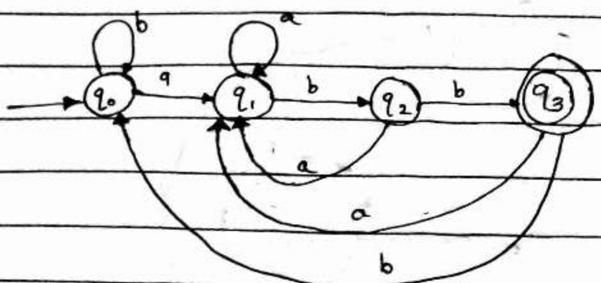
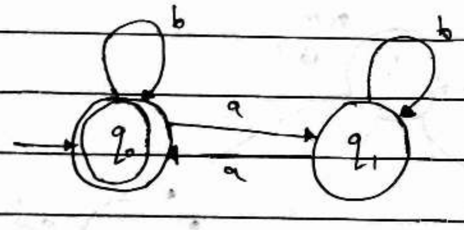
Q. DFA ends with 'abb'

abbabb
abbbabbb
abbb
abbaabb
abbbaabb

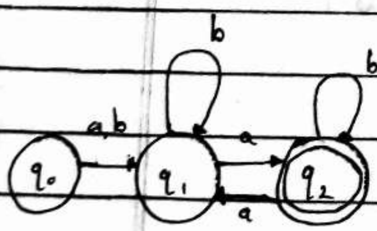


$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\delta : Q \times \Sigma \rightarrow Q$
$F = q_3$

Q. DFA for even no. of 'a's over $\Sigma = \{a, b\}$

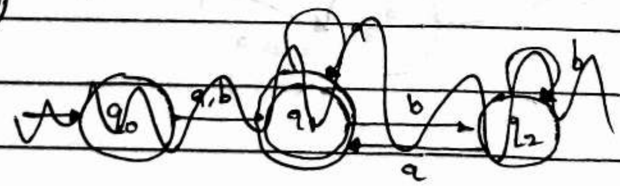


[ϵ is accepted]

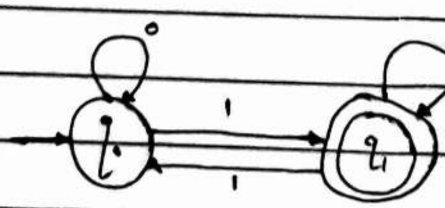


if there is $\Sigma^+ = \{a, b\}$
↳ nee closure

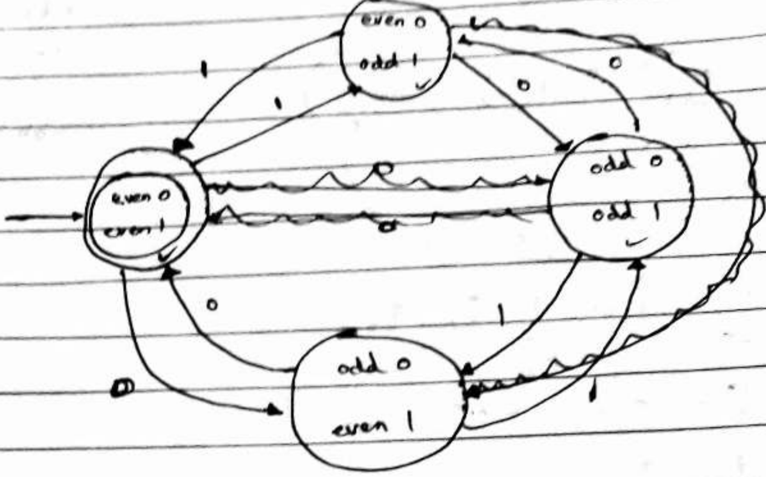
(i.e. $\Sigma^+ = \Sigma - \{\epsilon\}$)
↓
null value



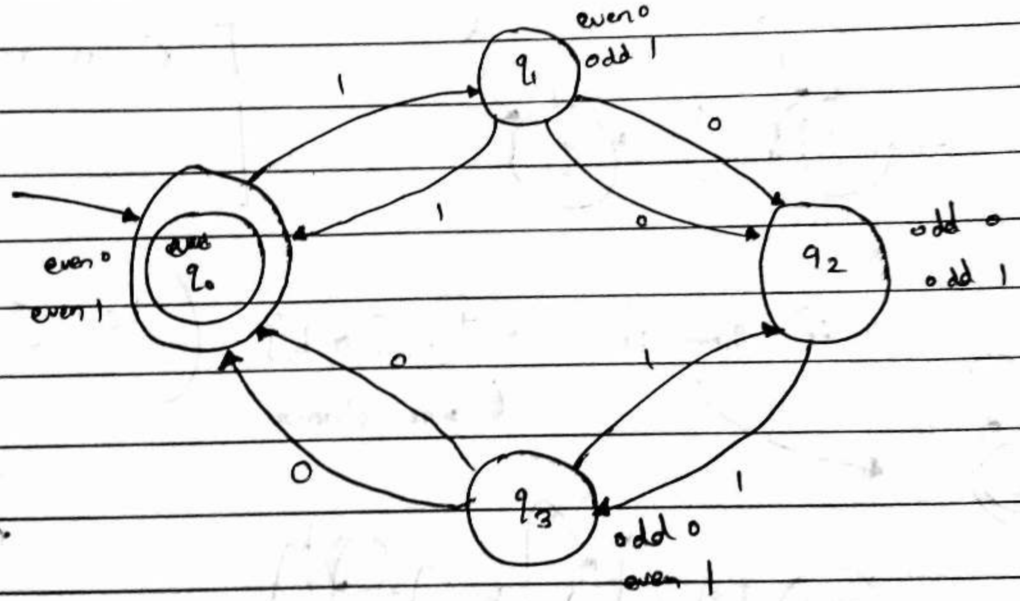
Q. DFA which accepts odd no. of '1's → $\Sigma = \{0, 1\}$



Q. Design DFA which accepts even 0's and even 1's $\Sigma = \{0, 1\}$



clean

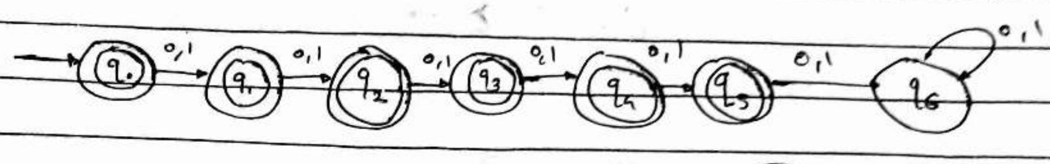


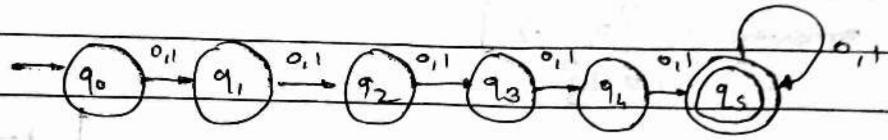
We are accepting ϵ

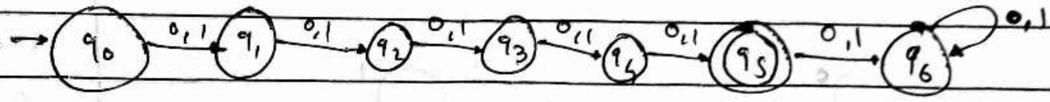
Q. Draw DFA for the following language $\Sigma = \{0, 1\}$

- length atmost 5
- length atleast 5
- length 5

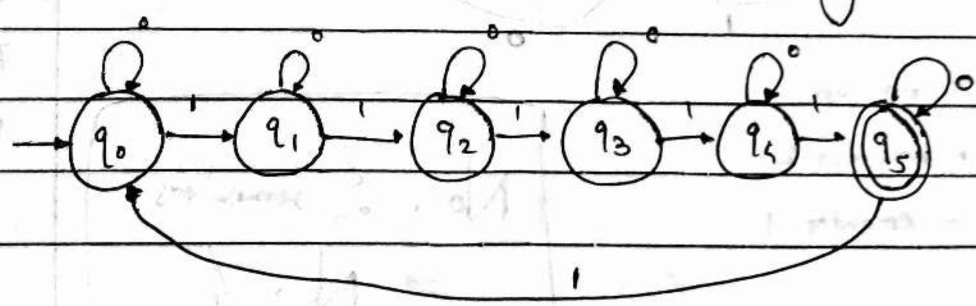
atmost 5



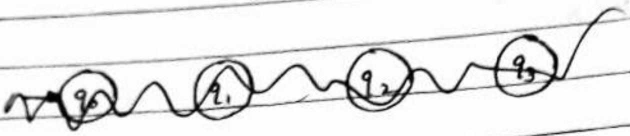




Q. Design DFA in which no. of ~~ones~~ 1's are divisible by 5.



Q. DFA which can accept no divisible by 3 (binary)

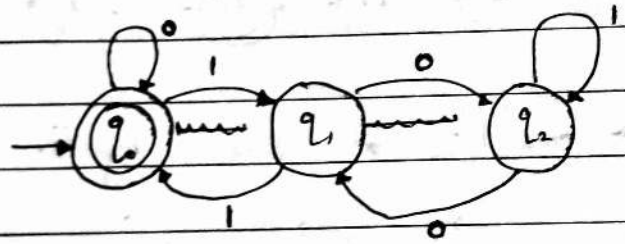


$$m = \{Q, \Sigma, q_0, \delta, F\}$$

$$\delta : Q \times \Sigma \rightarrow Q$$

remainders = $\{0, 1, 2\}$

$\Sigma = \{0, 1\}$



1 → remainder 1

10 → remainder 2

100 → remainder 1

11 → remainder 0

101 → remainder 2

$\boxed{\text{No. of remainders = No. of states}}$

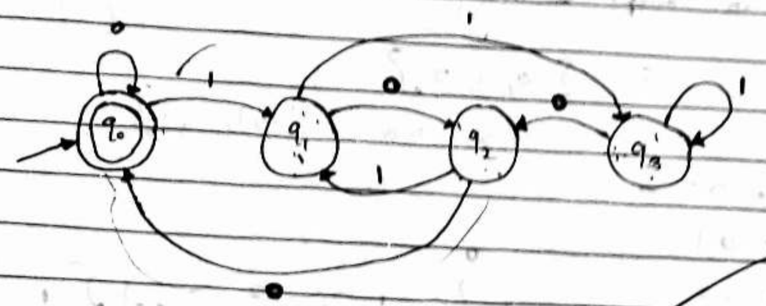
1 → 1
2 → 10
3 → 11
4 → 100
5 → 101
6 → 110
7 → 111
8 → 1000

In such problems

no. divisible by 4

remainders = {0, 1, 2, 3}

$\Sigma = \{0, 1\}$



1 → 1

10 → 2

11 → 3

100 → 0

101 → 1

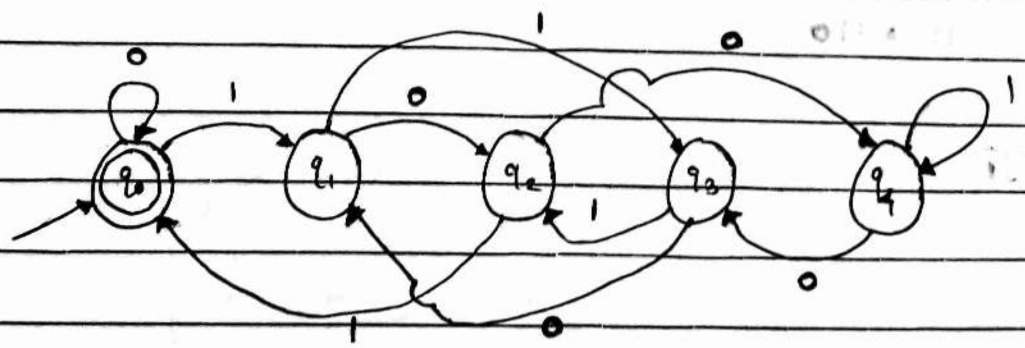
110 → 2

111 → 3

no. divisible by 5

remainders = {0, 1, 2, 3, 4}

$\Sigma = \{0, 1\}$



1 → 1

10 → 2

11 → 3

100 → 4

101 → 0

110 → 1

111 → 2

1000 → 3

1001 → 4

DATE / /

$m = (Q, \Sigma, q_0, \delta, F)$

$\delta : Q \times \Sigma \rightarrow Q$

Q. DFA which can accept ternary no. which is divisible by 4.

$\Sigma = \{0, 1, 2\}$

remainders $= \{0, 1, 2, 3\}$

$0 \rightarrow 00 \rightarrow 0$

$1 \rightarrow 01$

$2 \rightarrow 02$

$3 \rightarrow 10$

$4 \rightarrow 11$

$5 \rightarrow 12$

$6 \rightarrow 20$

$7 \rightarrow 21$

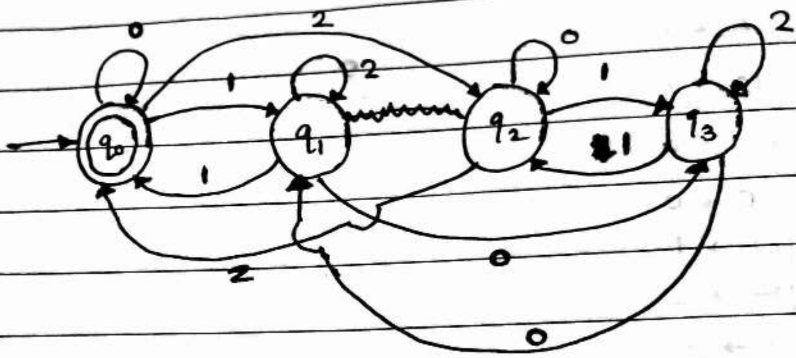
$8 \rightarrow 22$

$9 \rightarrow 100$

$10 \rightarrow 101$

$11 \rightarrow 102$

$12 \rightarrow 110$



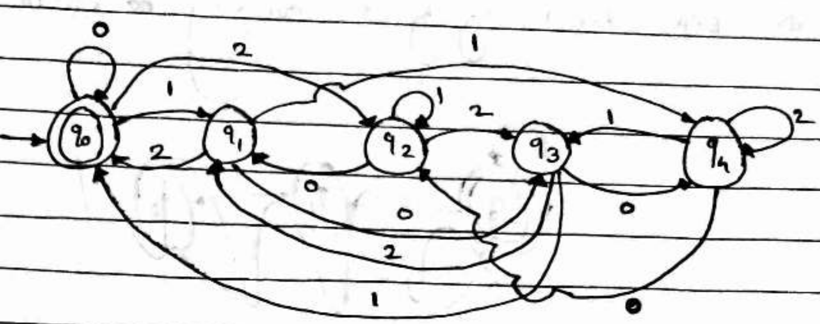
$4\overline{)1}$

Q. DFA which can accept ternary no. which is divisible by 5.

$\Sigma = \{0, 1, 2\}$

remainders = $\{0, 1, 2, 3, 4\}$

0 → 00
1 → 01
2 → 02
3 → 10
4 → 11
5 → 12
6 → 20
7 → 21
8 → 22
9 → 100
10 → 101
11 → 102
12 → 110
13 → 111
14 → 112
15 → 120

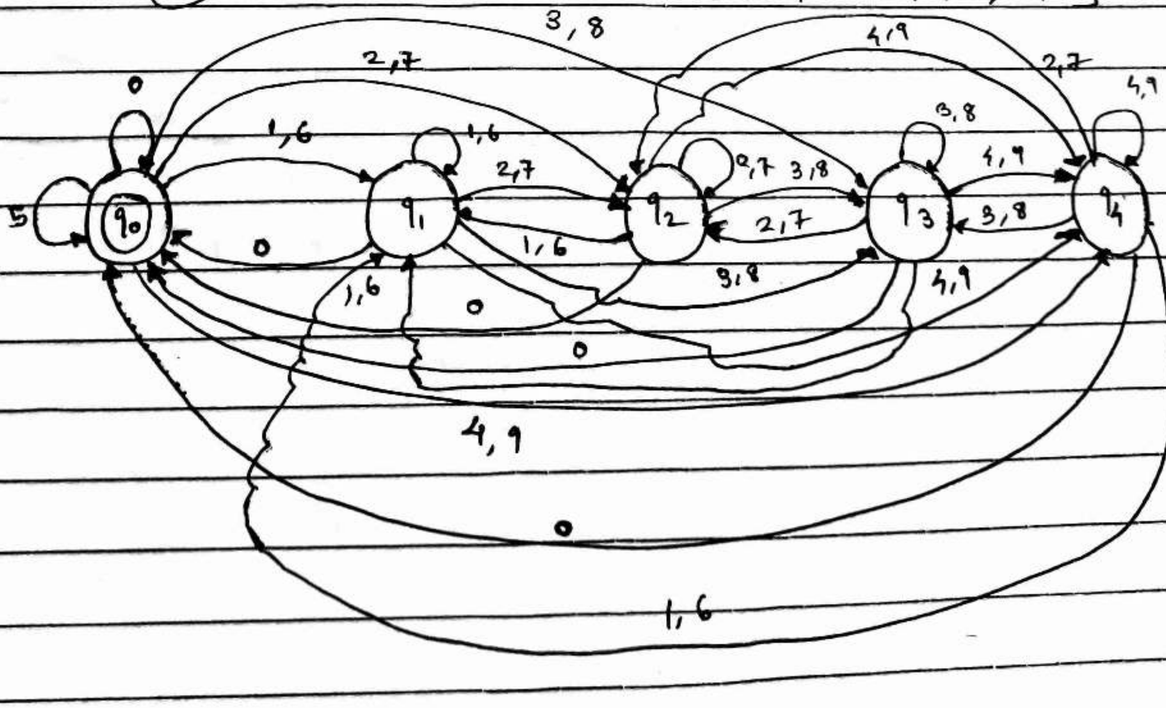


(mod 5)

Q FSM for divisibility 5 tester for given decimal no.

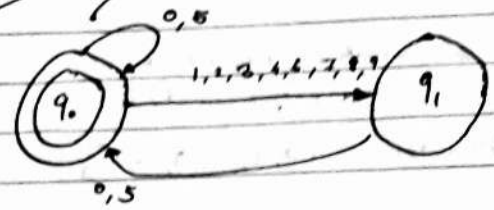
$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

~~0 → 00~~
1 → 01
2 → 02
3 → 03
4 → 04
5 → 05
6 → 06

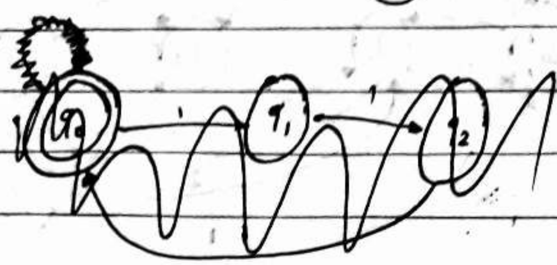


we have to find divisibility not remainder

elegant



Q. DFA for divisibility by 3 for unary no system

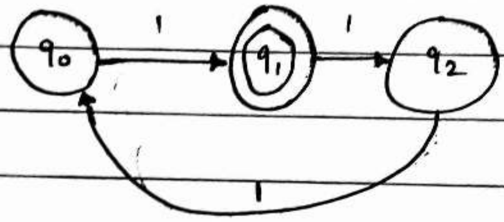


0 → 1

1 → ~~2~~ 11

2 → 111

3 → 1111



CR

Non deterministic finite automata

The language of a DFA: $A = (Q, \Sigma, q_0, \delta, F)$ is denoted by $L(A)$, & and defined by $L(A) = \{ w \mid \hat{\delta}(q_0, w) \rightarrow F \}$

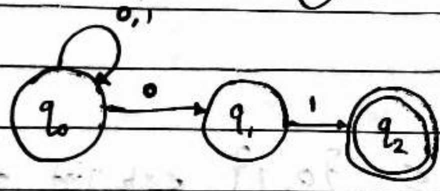
$w \rightarrow$ string

i.e. every string within the language starts from q_0 and reaches to the final state belongs to the language of A.

$A \rightarrow$ denotes no. of nodes

Non determinism property is defined as: for a given current state of a machine and an input symbol to read, next stage is not uniquely determined.

Q. NDFA $\Sigma = \{0,1\}$ where string ends with 01



	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	–	q_2
$*q_2$	–	–

1. more than one transition over one I/P
2. no transition
3. ϵ (null) transition

do not forget these in transition table

① $q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$ X

② $q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{0} \phi$ (null) $\xrightarrow{1} \phi$ X

③ $q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2$ ✓

A string is accepted by NDFA if there exists a single path that takes the machine to final state.

Power set of any set 's' is the set of all subsets of 's' including the empty set and 's' itself.
Denoted by 2^S.

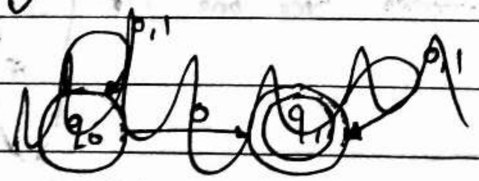
$$Q = (q_0, q_1, q_2)$$

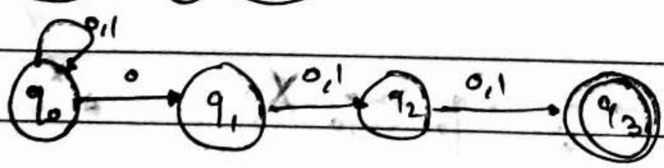
$$\{\epsilon, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0 q_1\}, \{q_1 q_2\}, \{q_2 q_0\}, \{q_0 q_1 q_2\}\}$$

$$m = (Q, \Sigma, q_0, \delta, F)$$

$$\delta : Q \times \Sigma \to \underbrace{2^Q}_{\text{one of the possible states}}$$

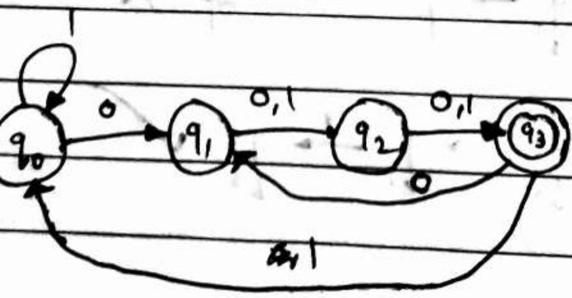
Q. NFA to accept a string over $\Sigma = \{0, 1\}$ such that third symbol from right side is 0.





1010

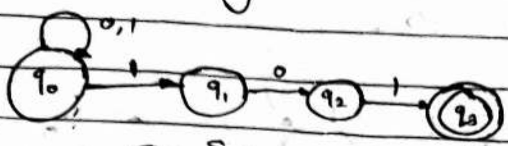
Q DFA from above



X wrong

0000

Q. NFA that accepts those strings which ends with 101.



$Q = \{q_0, q_1, q_2, q_3\}$

$\delta : Q \times \Sigma \rightarrow 2^Q$

$F = \{q_3\}$

$q_0 = \{q_0\}$

	0	1
q_0	q_0	q_0, q_1
q_1	q_2	—
q_2	—	q_3
q_3	—	—

$\delta(q_0, 1101)$

$\downarrow$

$\delta(q_0, 101 \cup q_1, 101)$

$\downarrow$

$\delta(q_0, 01 \cup q_1, 01 \cup \phi)$ — from q_1 there is no outgoing edge for 1, so ϕ.

$\downarrow$

$\delta(q_0, 1 \cup q_2 1)$

$\downarrow$

$\delta(q_0 \cup q_1 \cup q_3)$

as there is q_3 in the final state which is a part of F, hence this is a valid string.

Language L accepted by an NFA. $N = (Q, \Sigma, \delta, F, q_0)$ is defined as a set of all string accepted if it reaches the final state.

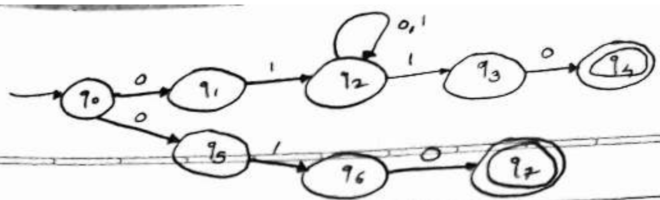
formally

$$L(m) = \{ \omega \in \Sigma^* \mid \hat{\delta}(q_0, \omega) \rightarrow F \}$$

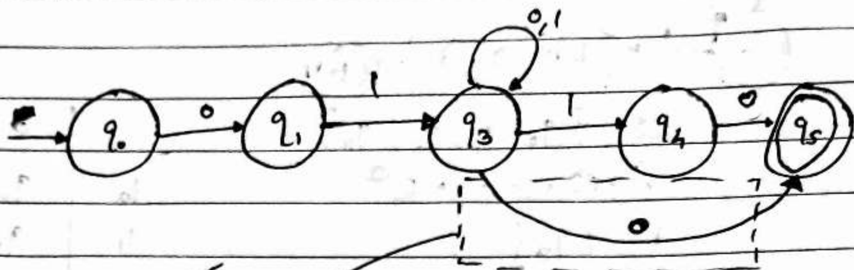
KNUTH
MORRIS
PRATT

PAGE NO.	
DATE	/ /

correct answer



Q. Design NFA for $\Sigma = \{0,1\}$ that starts with 01 and ends with 10.

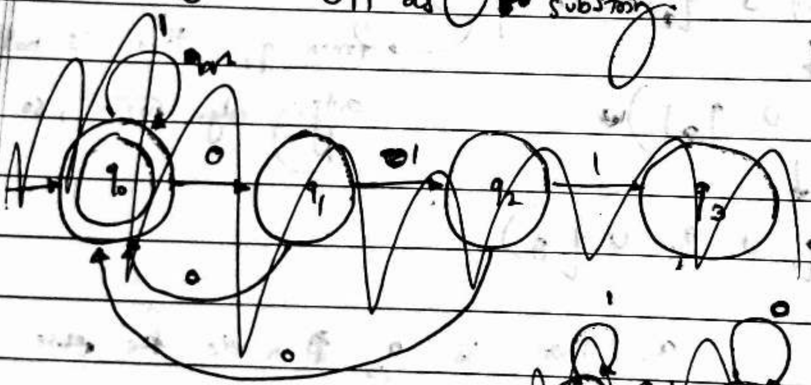


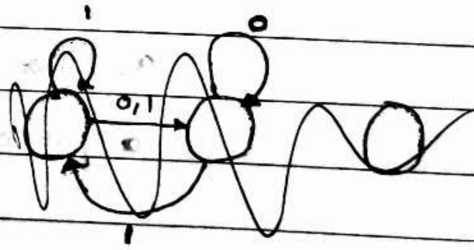
very interesting

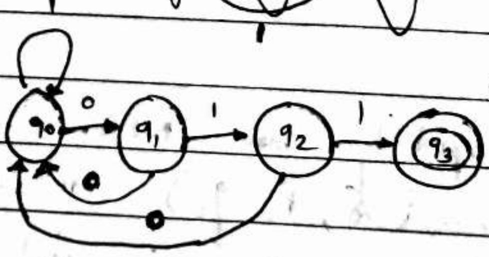
will accept 010 string (but will also accept 0100)

Q. Construct a NFA which accept string $\Sigma = \{0, 1\}$ accepting all possible string of 0,1 but does not contain 011 as substring.

very interesting

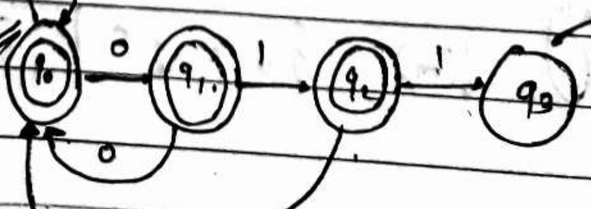






→ containing 011

no11



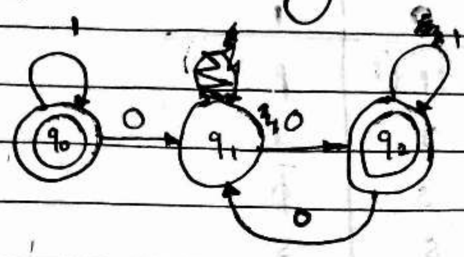
→ does not contain 011

Q. Design NFA $\Sigma = \{0,1\}$ consecutive two zeroes are allowed, single zero not allowed.

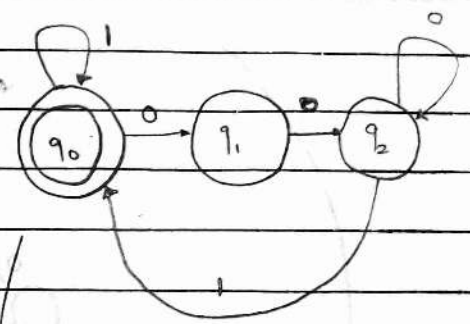
interesting

Q. Design NFA where first and last digit are same $\Sigma = \{a,b,c\}$

Q. Construct a machine that accept set of all string $\Sigma = \{a,b,c\}$ such that last symbol in input string also appears earlier in the string.

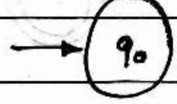


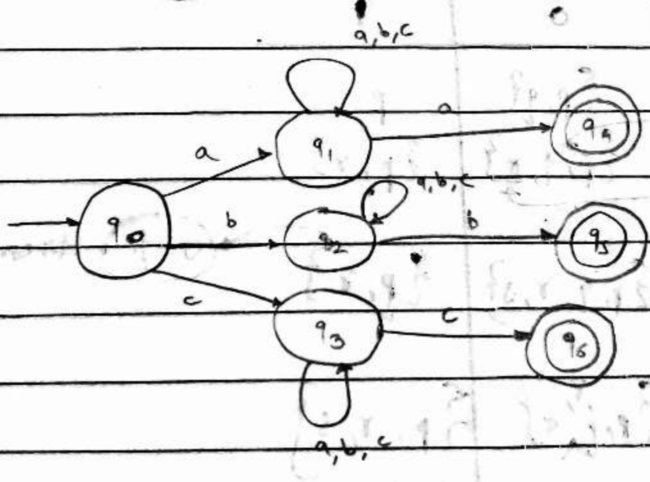
OR

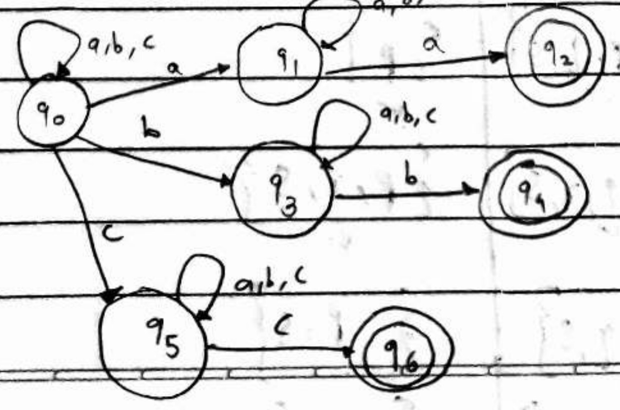


not sure

same







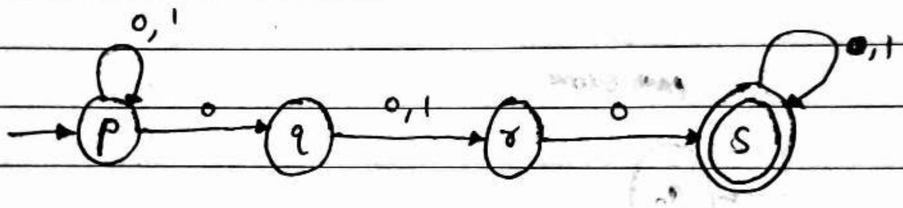
Subset construction

Q. Convert following NFA to equivalent DFA

$$M = \{(\{p, q, r, s\}, \Sigma = \{0,1\}, \delta, p, \{s\})\}$$

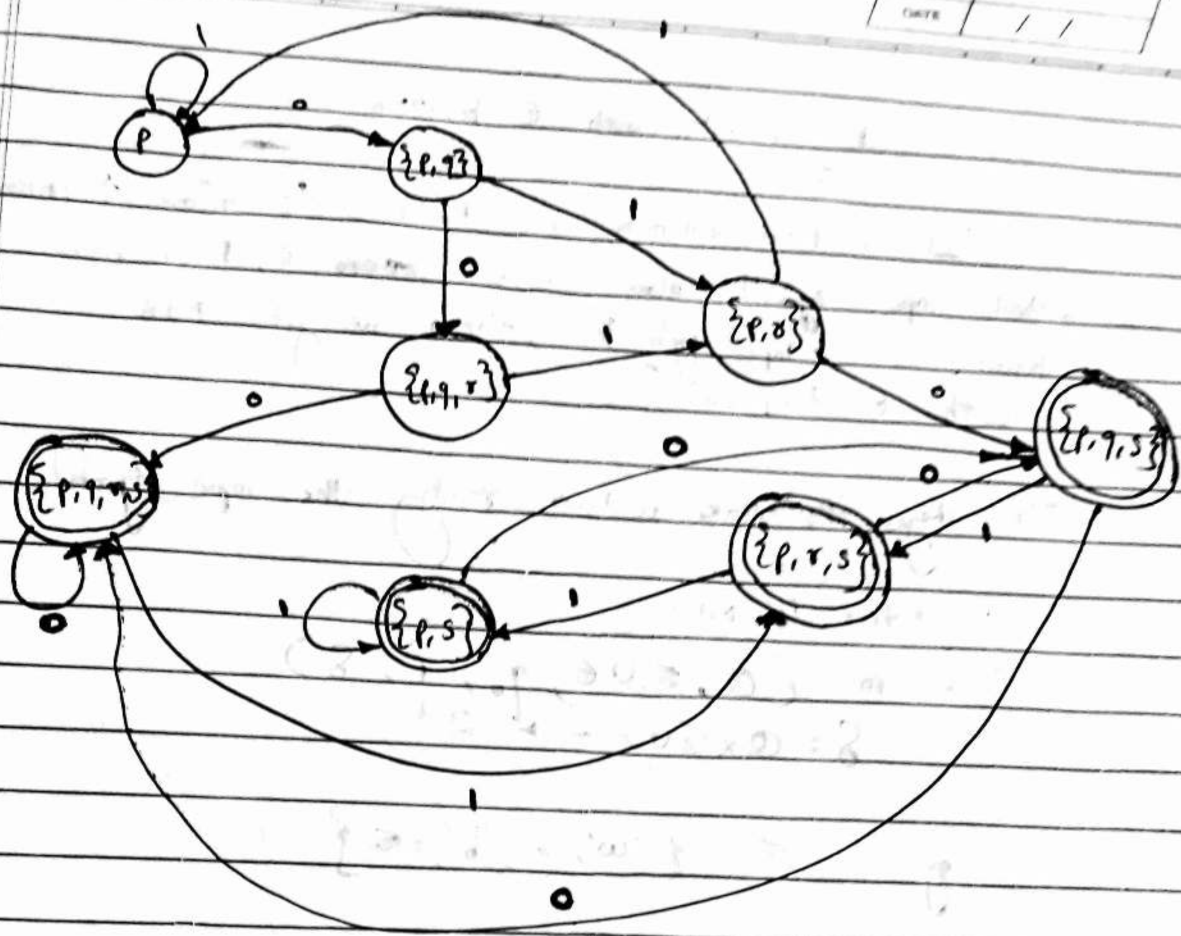
δ is given in the following table

	0	1
$\rightarrow p$	$\{p,q\}$	p
q	r	r
r	s	ϕ
$* s$	s	s



	0	1
p	$\{p,q\}$	p
$\{p,q\}$	$\{p,q,r\}$	$\{p,r\}$
$\{p,q,r\}$	$\{p,q,r,s\}$	$\{p,r\}$
$* \{p,q,r,s\}$	$\{p,q,r,s\}$	$\{p,r,s\}$
$\{p,r\}$	$\{p,q,s\}$	$\{p\}$
$* \{p,q,s\}$	$\{p,q,r,s\}$	$\{p,r,s\}$
$* \{p,r,s\}$	$\{p,q,s\}$	$\{p,s\}$
$* \{p,s\}$	$\{p,q,s\}$	$\{p,s\}$

$\cup\ \phi$ discarded


P
{P,q}
{P,r}
{P,q,r}
{P,q,r,s}
{P,q,s}
{P,r,s}
{P,s}
0
1

Finite automata with ϵ transition

If a finite automata is modified to allow transition without input symbol along with 0, 1 or more transition on input symbol, then we get NFA with ϵ transition.

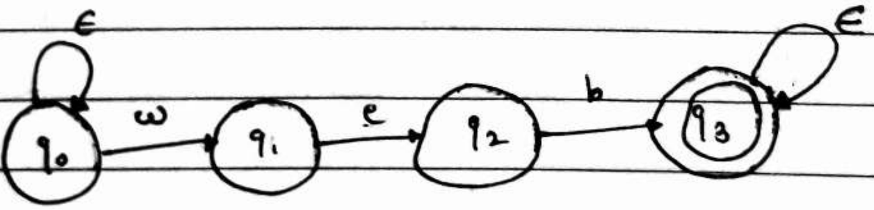
It changes the state without reading the input symbol.

for ϵ NFA

$$m = (Q, \Sigma \cup \epsilon, q_0, F, \delta)$$

$$\delta : Q \times \Sigma \cup \epsilon \rightarrow 2^Q$$

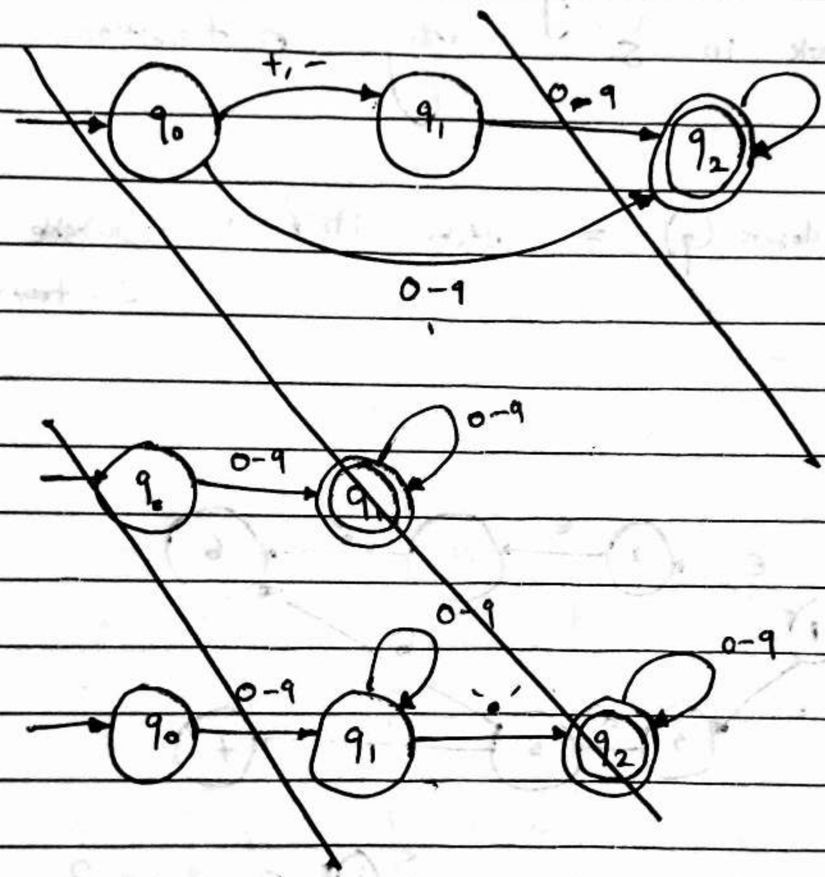
eg.- $\Sigma = \{ 'w', 'e', 'b', \epsilon \}$

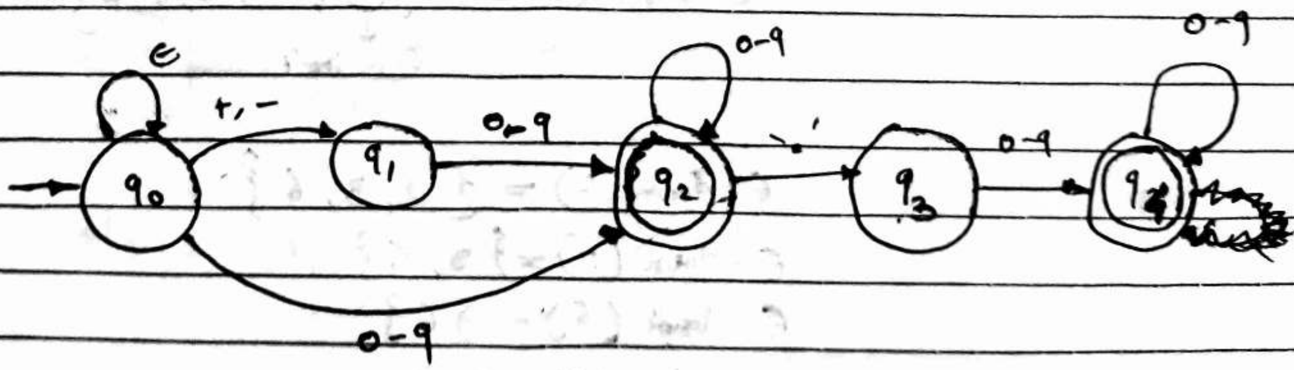


DATE / / /

Consider a ~~new~~ ε NFA that accepts decimal no. consisting of

1. an optional + or − sign
2. a string of digit
3. decimal point and string of digit.





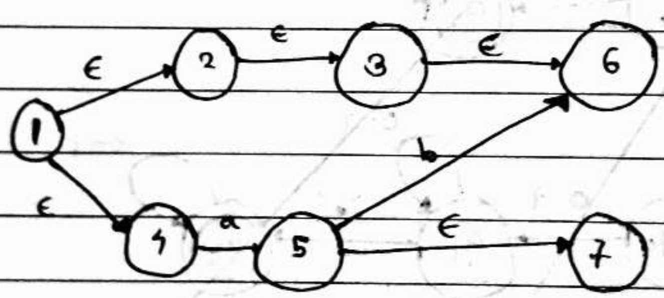
null NFA

epsilon closure

ε-NFA to DFA

ε-closure of state → of state Q denoted by ε-closure (q) is the set that contain q together with all the state that can be reached starting at q by following only ε transition. It will give set of state reachable from each state in S using ε transition.

ε-closure (q) = state itself + reachable states taking ε-transition.



ε-closure(1) = {1, 2, 3, 4, 6}

state itself

ε-closure(2) = { 2, 3, 6 }

ε-closure (3) = { 3, 6 }

ε-closure (6) = { 6 }

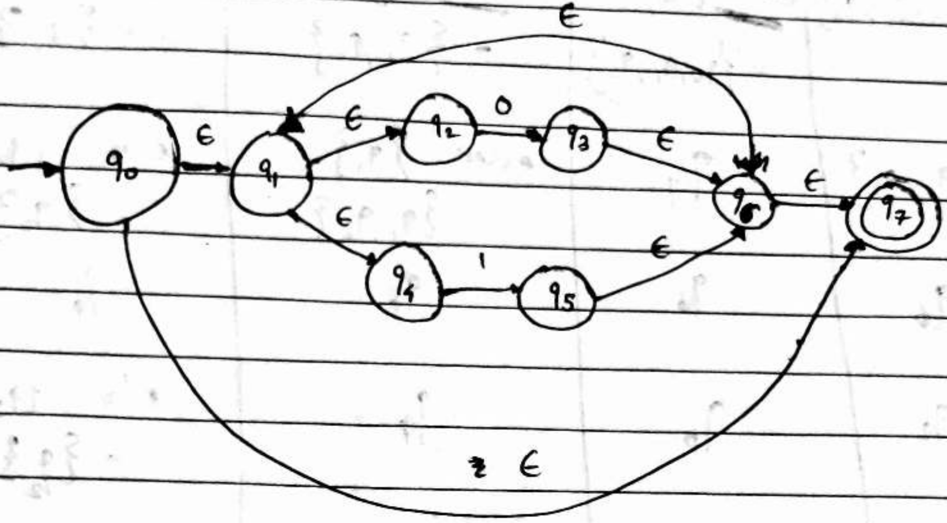
ε-closure (4) = { 4 }

ε-closure (5) = { 5, 7 }

ε-closure (7) = { 7 }

PAGE NO.
DATE / /

Calculate ϵ-closure of q_0, q_1, q_2, q_3, q_4



ϵ-closure (q_0) = $\{q_0, q_1, q_2, q_4, \cancel{q_6}, q_7\}$

ϵ-closure (q_1) = $\{q_1, q_2, q_4\}$

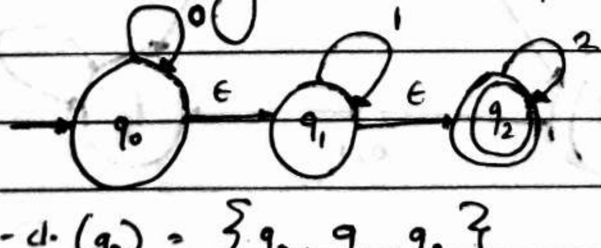
ϵ-closure (q_2) = $\{q_2\}$

ϵ-closure (q_3) = $\{q_3, q_6 \{, q_1, q_2, q_4\}, q_7\}$

ϵ-closure (q_4) = $\{q_4\}$

interesting

Convert the following ϵNFA to equivalent DFA

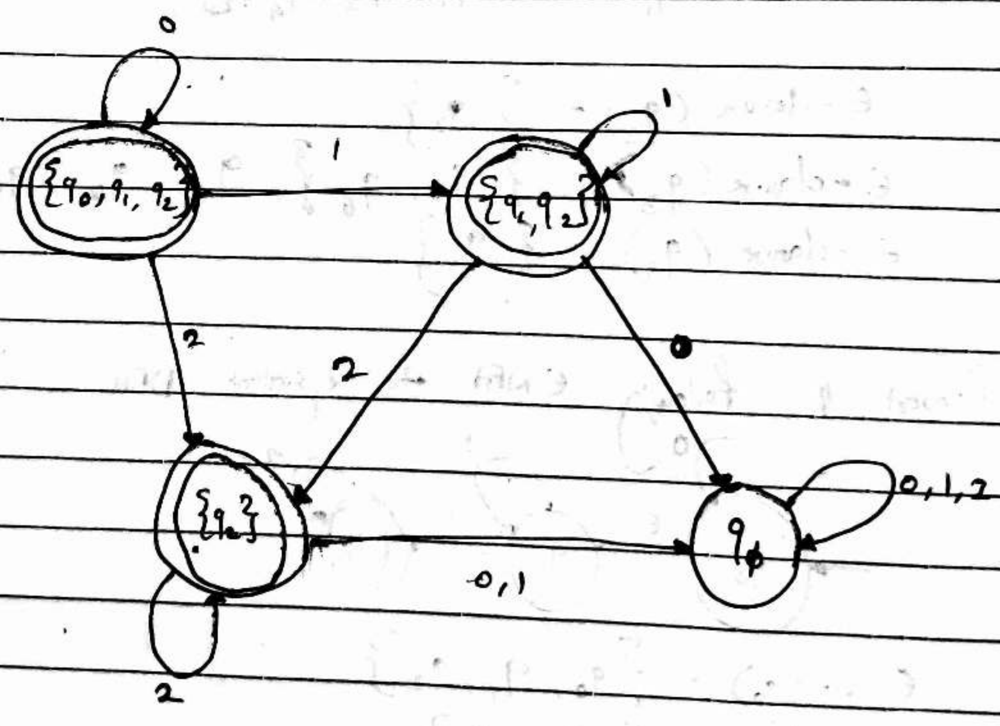


ϵ-cl.(q_0) = $\{q_0, q_1, q_2\}$

ϵ-cl.(q_1) = $\{q_1, q_2\}$

ϵ-cl.(q_2) = $\{q_2\}$

	0	1	2
$\rightarrow \{q_0, q_1, q_2\}$	e-closure $\{q_0\}$ $= \{q_0, q_1, q_2\}$	eclosure $\{q_1\}$ $= \{q_1, q_2\}$	eclosure $\{q_2\}$ $= \{q_2\}$
$\rightarrow \{q_1, q_2\}$	q_ϕ	eclosure $\{q_1\}$ $= \{q_1, q_2\}$	eclosure $\{q_2\}$ $= \{q_2\}$
q_ϕ	q_ϕ	q_ϕ	q_ϕ
q_2	q_ϕ	q_ϕ	eclosure $\{q_2\}$ $= \{q_2\}$



we can club them as $A = \{q_0, q_1, q_2\}$

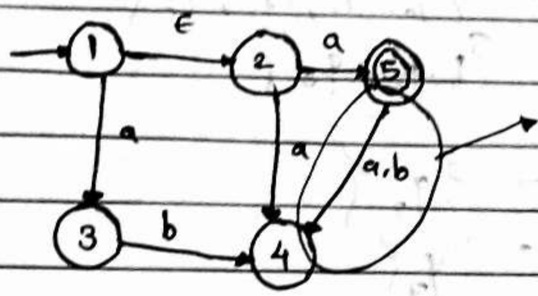
$B = \{q_1, q_2\}$

$C = \{q_2\}$

Algorithm

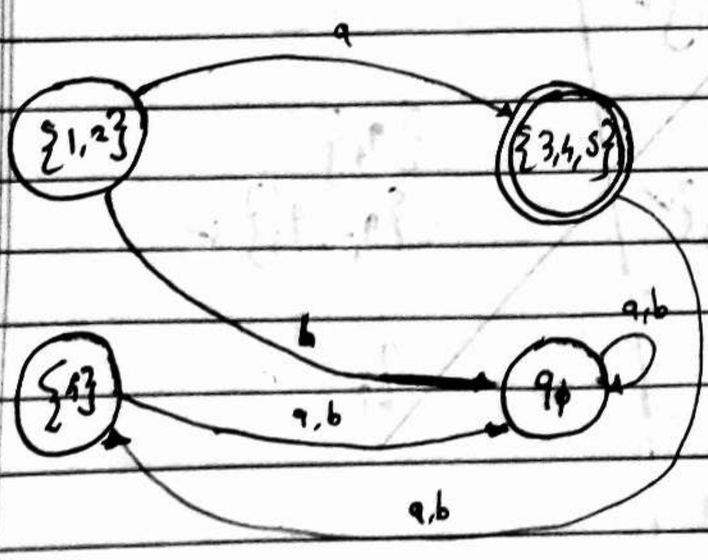
1. Take ε closure of beginning state of NFA as initial state of DFA.
2. Find states that can be traversed from the present for each input symbol and calculate ε closure of this state. This state becomes present state of DFA.
3. If any new state is found repeat step two.

Q. Find equivalent DFA for the ε - NFA

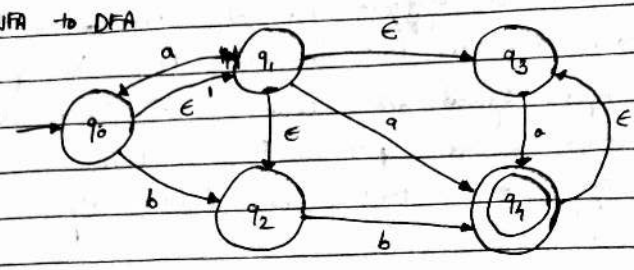


ϵ - clos (1) = {1, 2}
ϵ - cl. (2) = {2}
ϵ - cl. (3) = {3}.
ϵ - cl. (4) = {4}
ϵ - cl. (5) = {5}

	a	b
{1, 2}	ε-closure {3, 5}{4} = {3, 4, 5}	{q_ϕ}
{3, 4, 5}	ε-closure {4} = {4}	ε-closure {5} = {5}
{4}	{q_ϕ}	{q_ϕ}
{q_ϕ}	{q_ϕ}	{q_ϕ}



Q. ϵ-NFA to DFA



$\epsilon\text{-cl.}(q_0) = \{q_0, q_1, q_2, q_3\}$

$\epsilon\text{-cl.}(q_1) = \{q_1, q_2, q_3\}$

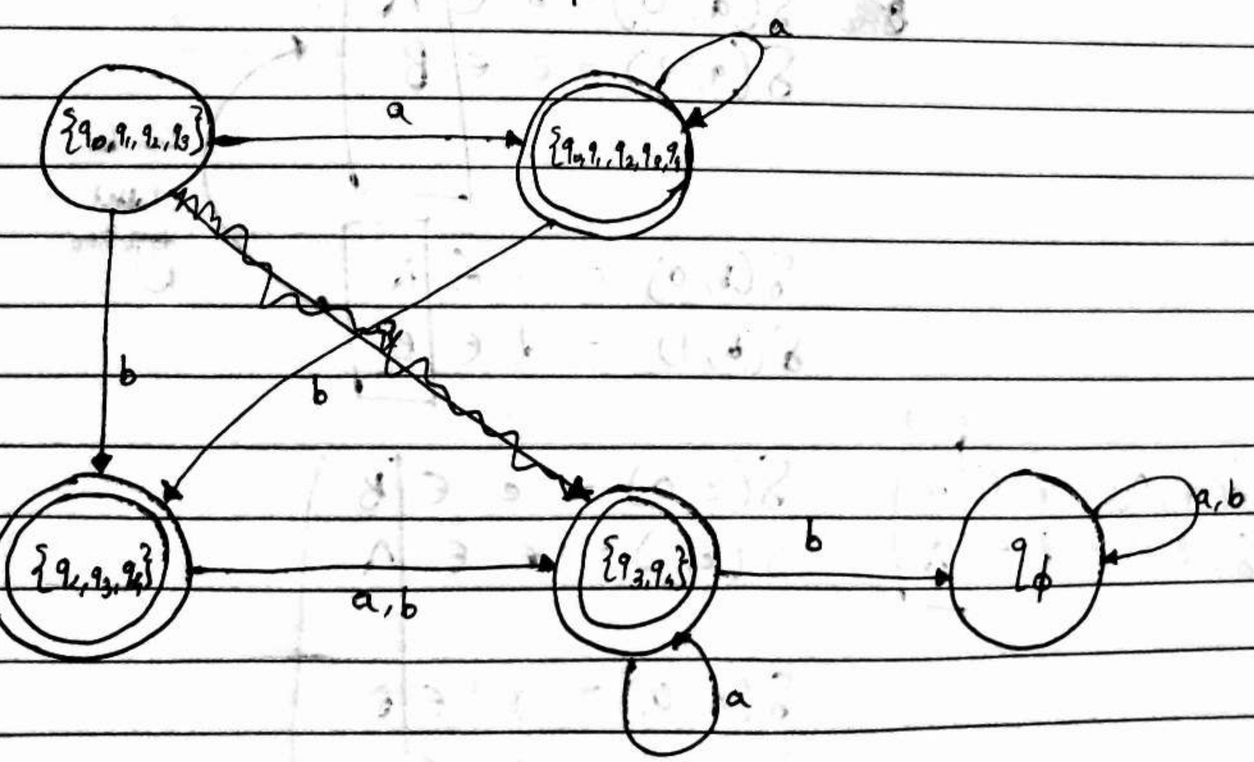
$\epsilon\text{-cl.}(q_2) = \{q_2\}$

$\epsilon\text{-cl.}(q_3) = \{q_3\}$

$\epsilon\text{-cl.}(q_4) = \{q_3, q_4\}$

	a	b
$\{q_0, q_1, q_2, q_3\}$	$\epsilon\text{-cl.}\{q_3, q_4\} = \{q_3, q_4\}$	$\epsilon\text{-cl.}\{q_2, q_4\} = \{q_2, q_3, q_4\}$
$\{q_3, q_4\}$	$\{q_4\} = \{q_3, q_4\}$ ✓	$\{q_\phi\}$
$\{q_2, q_3, q_4\}$	$\{q_4\} = \{q_3, q_4\}$ ✓	$\{q_4\} = \{q_3, q_4\}$ ✓
q_ϕ	q_ϕ	q_ϕ

	a	b
$\{q_0, q_1, q_2, q_3\}$	e-cl. $\{q_0, q_4\}$ $= \{q_0, q_1, q_2, q_3, q_4\}$	e-cl. $\{q_2, q_4\}$ $= \{q_2, q_3, q_4\}$
$\{q_0, q_1, q_2, q_3, q_4\}$	$\{q_0, q_4\}$ $= \{q_0, q_1, q_2, q_3, q_4\}$ ✓	e-cl. $\{q_2, q_4\}$ $= \{q_2, q_3, q_4\}$
$\{q_2, q_3, q_4\}$	$\{q_4\}$ $= \{q_3, q_4\}$	$\{q_4\}$ $= \{q_3, q_4\}$
$\{q_3, q_4\}$	$\{q_4\}$ $\{q_3, q_4\}$	$\{q_\phi\}$
$\{q_\phi\}$	$\{q_\phi\}$	$\{q_\phi\}$



DATE / /

Minimization of DFA

Equivalence Theorem

States	0	1
→ a	b	c
b	a	d
* c	e	f
* d	e	f
* e	e	f
f	f	f

P_0 $\quad A = \{a, b, f\}$ $\qquad B = \{c, d, e\}$

N.F (non final)

$\delta(a, 0) = b \in A$

$\delta(a, 1) = c \in B$

can be clubbed together

$\delta(b, 0) = a \in A$

$\delta(b, 1) = d \in B$

$\delta(f, 0) = f \in A$

$\delta(f, 1) = f \in A$

$\delta(c, 0) = e \in B$

$\delta(c, 1) = f \in A$

$\delta(d, 0) = e \in B$

$\delta(d, 1) = f \in A$

$\delta(e, 0) = e \in B$

$\delta(e, 1) = f \in A$

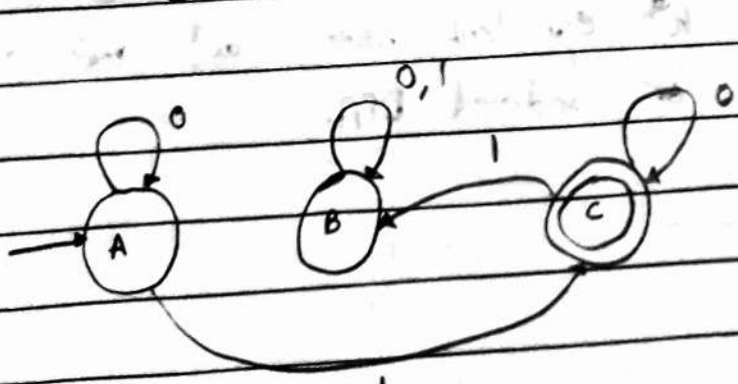
where $A = \{a, b\}$
$C = \{c, d, e\}$
$B = \{f\}$

$A = \{a, b\}$ $\quad B = \{f\}$
$C = \{c, d, e\}$

$\delta(a, 0) = b \in A$
$\delta(a, 1) = c \in C$
$\delta(b, 0) = b \in A$
$\delta(b, 1) = d \in C$

$\delta(c, 0) = e \in C$
$\delta(c, 1) = f \in B$
$\delta(d, 0) = e \in C$
$\delta(d, 1) = f \in B$
$\delta(e, 0) = e \in C$
$\delta(e, 1) = f \in B$

$\delta(f, 0) = f \in B$
$\delta(f, 1) = f \in B$



very interesting

when we convert NFA to DFA, check?
we get a minimised DFA → this is not a theorem, just observation
still convert a minimised DFA

[interesting observation]

Equivalence Theorem examples

if 'x' and 'y' are two states in DFA we can combine these two states into $\{x, y\}$ if they are identical.

Two states are identical if there is atleast one string 's' such that transition f^n $\delta(x, s)$ and $\delta(y, s)$ is accepting any other is also in accepting state.

Algorithm :-

step 1 : All the states Q are devided into two :-
non final and final state
denoted by P_0.

step 2 : For each partition in P_k, divide the state into two partition if they are k^{th} distinguishable.

Step 3: Combine the k^{th} equivalent state and make the new state of reduced DFA.

Q.

	0	1
→A	B	E
B	C	F
C	D	H
D	E	H
E	F	I
F*	G	B
G	H	B
H	I	C
I*	A	E

$P_0 = \overbrace{\{A, B, C, D, E, G, H\}}^{AA} \quad \overbrace{\{F, I\}}^{BB}$

$\delta(A,0) = B \in AA$
$\delta(A,1) = E \in AA$

$\delta(B,0) = C \in AA$
$\delta(B,1) = F \in BB$

$\delta(C,0) = D \in AA$
$\delta(C,1) = H \in AA$

Do not ~~merge~~ final and non final states

We do not merge any sets ~~on~~

$\delta(D,0) = E \in AA$
$\delta(D,1) = H \in AA$

$\delta(H,0) = I \in BB$
$\delta(H,1) = C \in AA$

$\delta(E,0) = F \in BB$
$\delta(E,1) = I \in BB$

F $\delta(I,0) = A \in AA$
$\delta(I,1) = E \in AA$

F $\delta(F,0) = G \in AA$
$\delta(F,1) = B \in AA$

$\delta(G,0) = H \in AA$
$\delta(G,1) = B \in AA$

$P_1 =$ non final $\overbrace{\{A, C, D, G\}}^{AA}, \overbrace{\{B\}}^{BB}, \overbrace{\{E\}}^{CC}, \overbrace{\{H\}}^{DD}, \overbrace{\{F, I\}}^{EE}$

$\delta(A,0) = B \in BB$
$\delta(A,1) = E \in CC$

$\delta(C,0) = D \in AA$
$\delta(C,1) = H \in DD$

$\delta(D,0) = E \in CC$
$\delta(D,1) = H \in DD$

$\delta(G,0) = H \in DD$
$\delta(G,1) = B \in BB$

F $\delta(F,0) = G \in AA$
$\delta(F,1) = B \in BB$

$\delta(I,0) = A \in AA$
$\delta(I,1) = E \in CC$

$P_2 = \{A\} \{B\} \{C\} \{D\} \{E\} \{F\} \{G\} \{H\} \{I\}$

PAGE NO.
DATE / /

finite automata with output

Q.

	ϵ	a	b	c
$\rightarrow$ p	$\{p,q,r\}$	ϕ	$\{q\}$	$\{r\}$
q	ϕ	$\{p\}$	$\{r\}$	$\{p,q\}$
* r	ϕ	ϕ	ϕ	ϕ

Consider the following finite machine part A.
compute the epsilon closure of each state.
find out the subset construction and equivalence theorem gives same output or not.

$$\downarrow \text{subset construction}$$

DFA_1

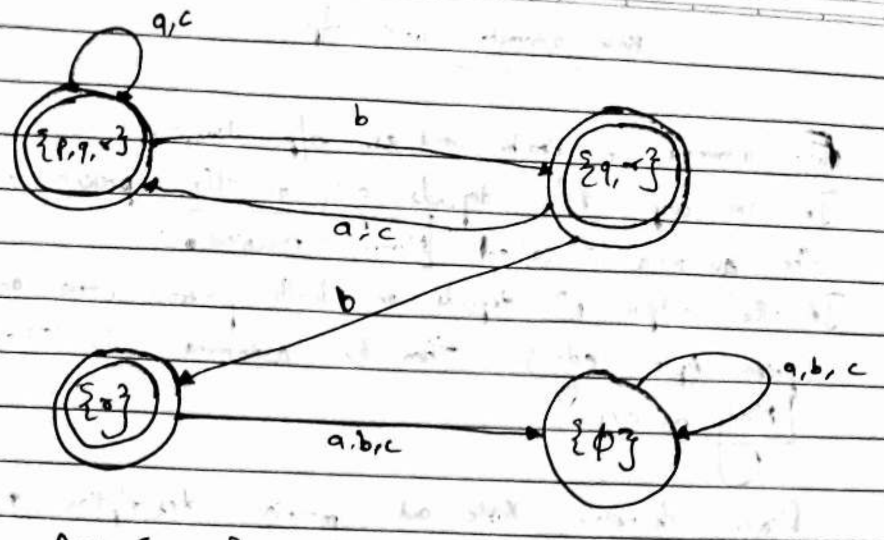
$$\downarrow \text{equivalence theorem}$$

DFA_2

ϵ-cl. $(p) \rightarrow \{p,q,r\}$
ϵ-cl. $(q) \rightarrow \{q\}$
ϵ-cl. $(r) \rightarrow \{r\}$

interesting

	a	b	c
	ϵ-cl.	ϵ-cl.	ϵ-cl.
* $\{p,q,r\}$	$\{p\}$	$\{q,r\}$	$\{p,q,r\}$
	$\{p,q,r\}$	$\{q,r\}$	$\{p,q,r\}$
* $\{q,r\}$	$\{p\}$	$\{r\}$	$\{p,q\}$
	$\{p,q,r\}$	$\{r\}$	$\{p,q,r\}$
* $\{r\}$	$\{\phi\}$	$\{\phi\}$	$\{\phi\}$
$\{\phi\}$	$\{\phi\}$	$\{\phi\}$	$\{\phi\}$



$A = \{p, q, r\}$

$B = \{q, r\}$

$C = \{r\}$

$P_0 = \underbrace{\{A, B, C\}}_{Q} \quad \underbrace{\{\phi\}}_{R}$

$\delta(A, a) = \{p, q, r\}\ (A) \in Q$

$\delta(A, b) = \{q, r\}\ (B) \in Q$

$\delta(A, c) = \{p, q, r\}\ (A) \in Q$

$\delta(B, a) = \{p, q, r\}\ (A) \in Q$

$\delta(B, b) = \{r\}\ (C) \in Q$

$\delta(B, c) = \{p, q, r\}\ (A) \in Q$

$\delta(C, a) = \{q_\phi\} \in R$

$\delta(C, b) = \{q_\phi\} \in R$

$\delta(C, c) = \{q_\phi\} \in R$

$P_1 = \underbrace{\{A, B\}}_{Q}, \underbrace{\{C\}}_{R}, \underbrace{\{q_\phi\}}_{S}$

$\delta(A, a) = \{p, q, r\} \in Q$	$\delta(B, a) = \{p, q, r\} \in Q$
$\delta(A, b) = \{q, r\} \in Q$	$\delta(B, b) = \{r\} \in R$
$\delta(A, c) = \{p, q, r\} \in Q$	$\delta(B, c) = \{p, q, r\} \in Q$

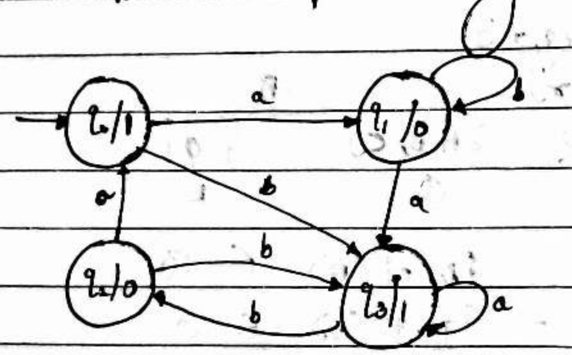
$P_2 = \{A\}\ \{B\}\ \{C\}\ \{q_\phi\}$ (No minimization)

DATE / /

Finite automata with o/p

Finite automata can also be used as o/p device.
If the output f^n depends only on the present state, the automata is called **Moore machine**.
If the output f^n depends on both present state and present input (edge) then the automata is called **Mealy machine**.

Design transition table and machine description for given figure and find the output for string 'abab'.



10010

	a	b	o/p
q_0	q_1	q_3	1
q_1	q_3	q_1	0
q_2	q_0	q_3	0
q_3	q_3	q_2	1

$\omega = abab$

$\boxed{1}0010$

we have to count the initial state's output
books follow two thoughts

- ignore the initial state's output
- substitute it with a 'ignore' character

000100

$w = bbba$

11011

000100

$m = (Q, \Sigma, O, q_0, \delta, \lambda)$

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{a, b\}$

O = output symbols = $\{0, 1\}$

$\delta : Q \times \Sigma \rightarrow Q$ (transition f^n)

$\lambda : Q \rightarrow O$ (output f^n)

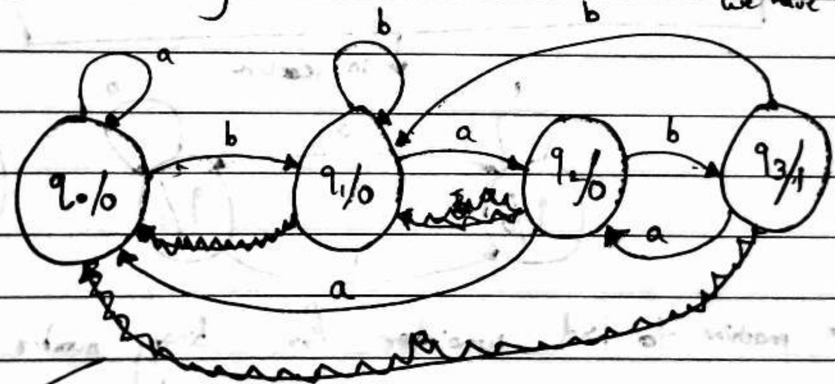
last bit should be

Q. Design a Moore machine that gives an output 1, if input string ends with bab.

first all are 0. — not given in Q, we have to assume

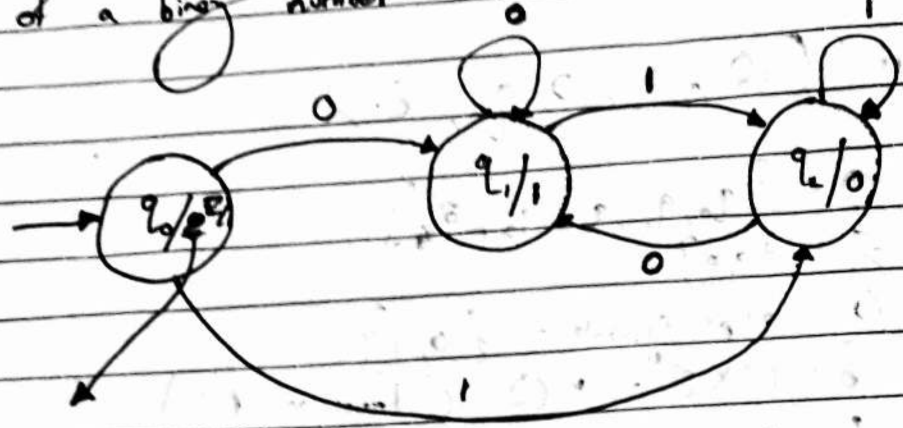
$\Sigma = \{a, b\}$

$O = \{0, 1\}$



very interesting — We check the last bit after the input is parsed, if the last bit is 1, then it ends with bab

Q. Design a moore machine for the 1's complement of a binary number.



this start state will not print anything

(we are ignoring initial state's output.)

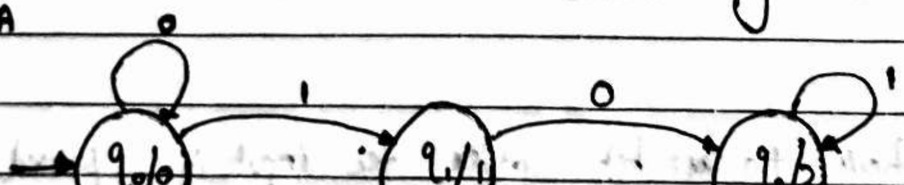
in lecture

(So start with 0)

Q. Moore machine to find remainder for binary number for 3.

000	
001	1
010	2
011	3
100	4
101	5
110	6

DFA

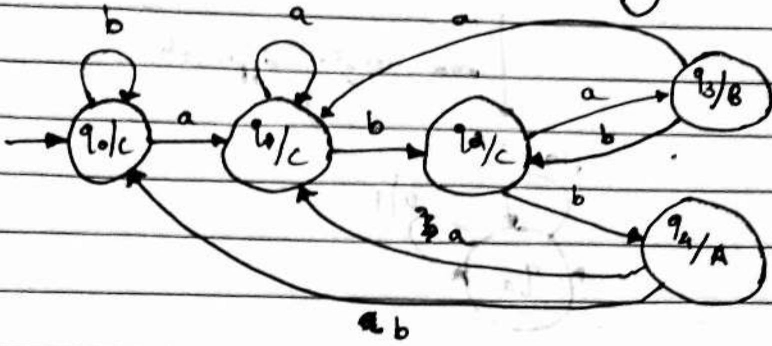


PAGE NO / DATE / /

Q. moore machine to generate output A if string is ending with 'abb'.
B if string is ending with 'aba'.
C if string is ending otherwise

$\Sigma = \{a, b\}$

$O = \{A, B, C\}$



abaa

abbb

abbabb

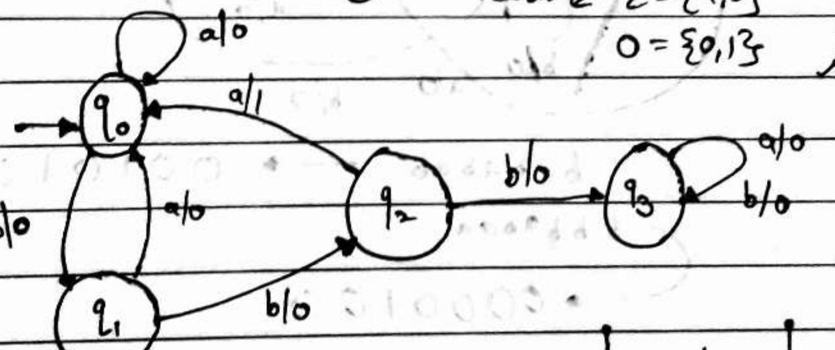
Q. Design a transition table machine desc. of a given fig.
find the output of this string.

$m = (Q, \Sigma, O, \delta, \lambda, q_0)$

$\Sigma = \{a, b\}$ $\quad \delta : Q \times \Sigma \to Q$

$O = \{0, 1\}$ $\quad \boxed{\lambda : Q \times \Sigma \to O}$

interesting



	a	o/p	b	o/p
q_0	q_0	0	q_1	0
q_1	q_0	0	q_2	0
q_2	q_0	1	q_3	0
q_3	q_3	0	q_3	0

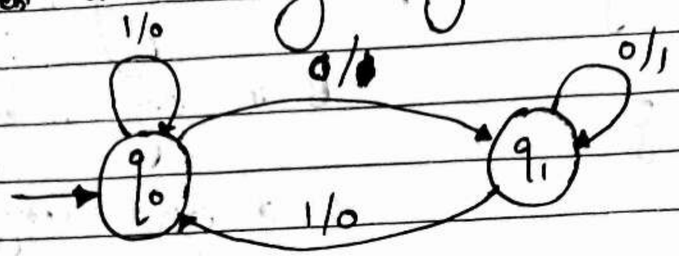
ω = bbab

0010

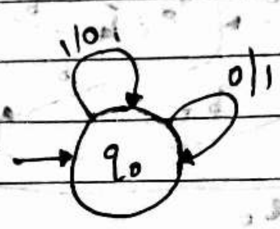
ω = abaab

00000

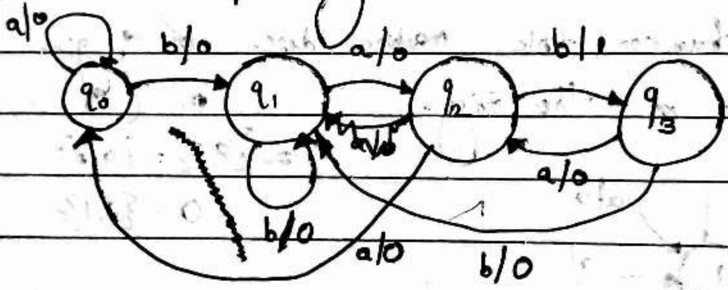
Q. 1's complement of a no. using Mealy machine.



more interesting approach



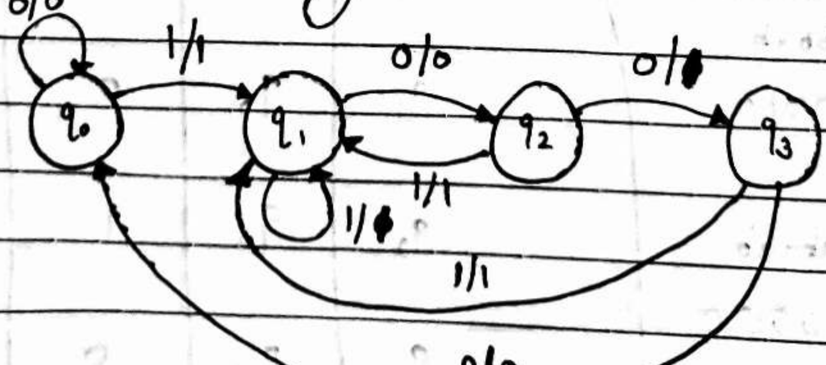
Q. gives output 1 if input string ends in 'bab'



babab ab ⟶ 0010101

bbbabaa ⟶ 0000100

Q. Converts each occurrence of substring 100 to 101



1000

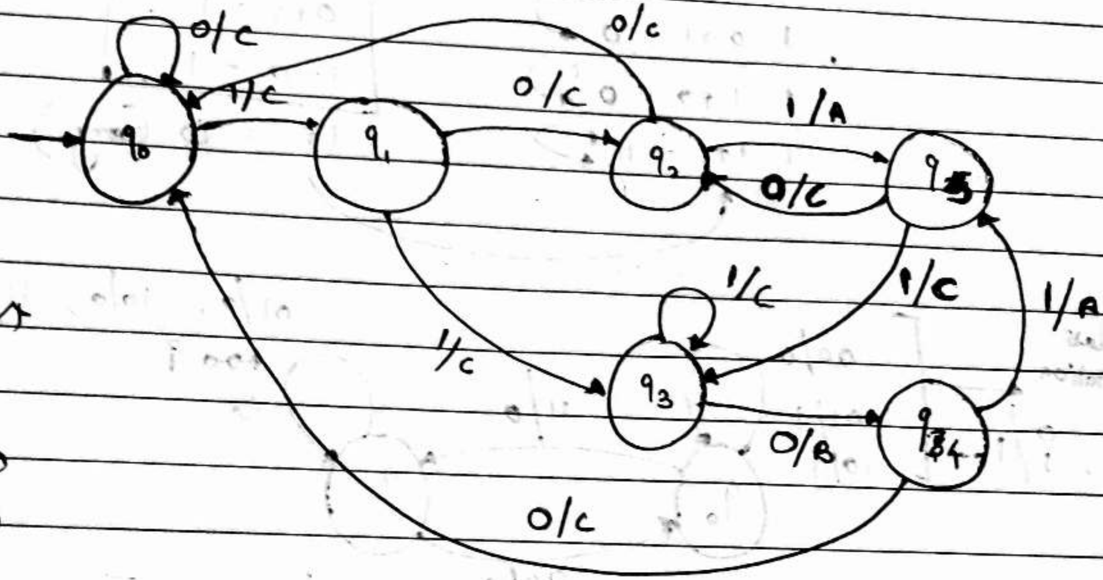
1000100

PAGE NO.
DATE / /

mealy m/c

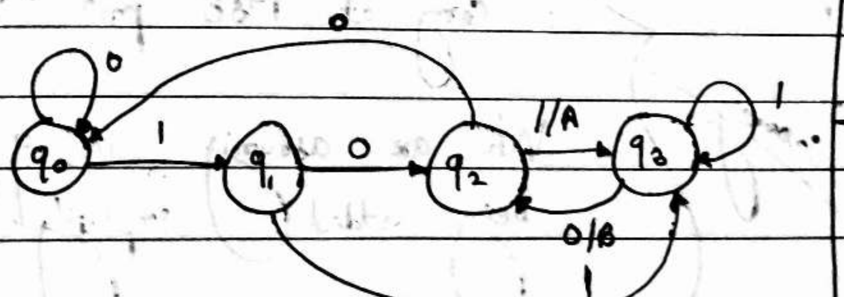
Q. Design a mealy machine that accepts an input from $(0+1)^*$.
if input ends in 101 → output is A
110 → output is B
else C

$(0+1)^*$ → any combination of 0 and 1



1000
1110
1010 1
1010 1
1011
1101
1101
101 110
A
10110
1010

Toray's method



This is wrong! invalid for $\boxed{1010}$

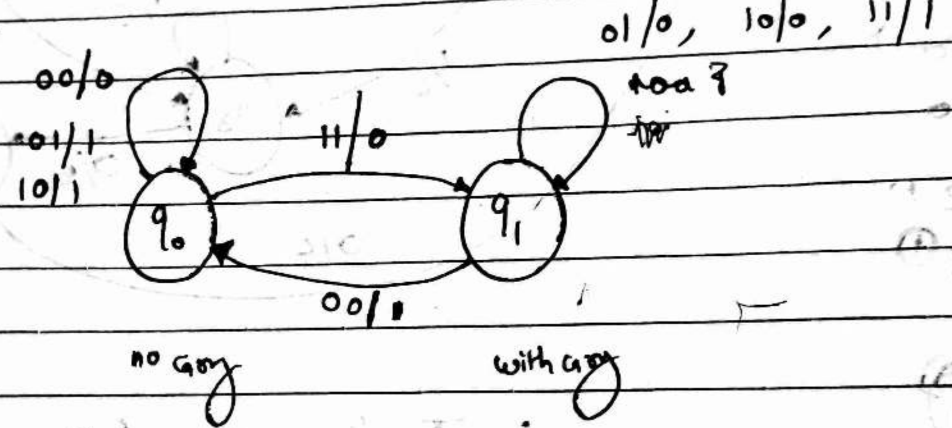
Q. Design a mealy machine to find addition of two binary no.
(interesting)

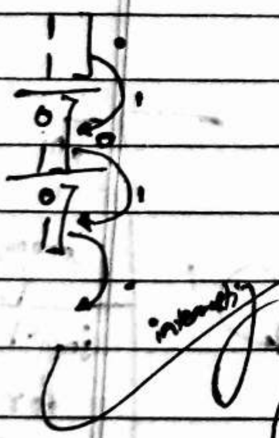
carry	
1	0+0 = 1
1	0+1 = 0
1	1+0 = 0
1	1+1 = 1

no carry
0+0 = 0
0+1 = 1
1+0 = 1
1+1 = 0 (carry)

in class notation

$\frac{0}{0}/0$, $\frac{0}{1}/1$ ...





If the ans ends in q_1 state, carry at MSB pos^n is seen.

We are assuming in q_1 state carry is being added implicitly.

Q. Design a mealy machine to find substraction of two binary no.

borrow — no borrow

1 - 0 - 0 = 1

1 - 0 - 1 = 0

$\boxed{\begin{array}{l} 0 - 0 = 0 \\ 1 - 0 = 1 \\ 1 - 1 = 0 \end{array}}$

0 - 1 = 1

100
100

110
101
0001

110 → 6
101 → 5

Equivalence of Mealy and Moore machines

Moore ⇄ Mealy

Moore to Mealy state

	0	1	o/p
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

→

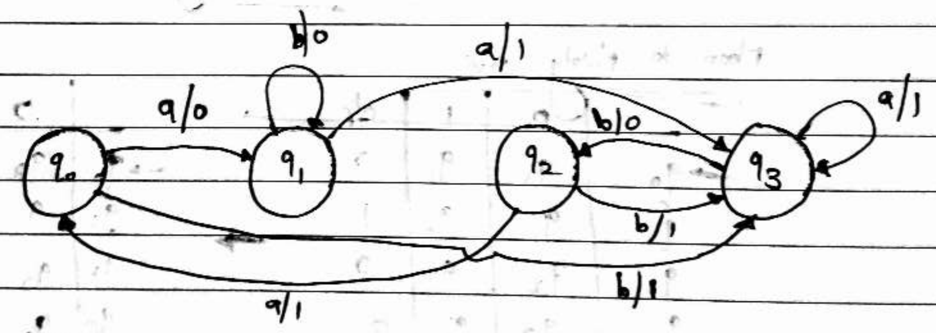
	0	o/p	1	o/p
→ q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

0 110
1 0 0 0
1 0 1 0
---
1 0 0

Q. convert this Moore to Mealy machine

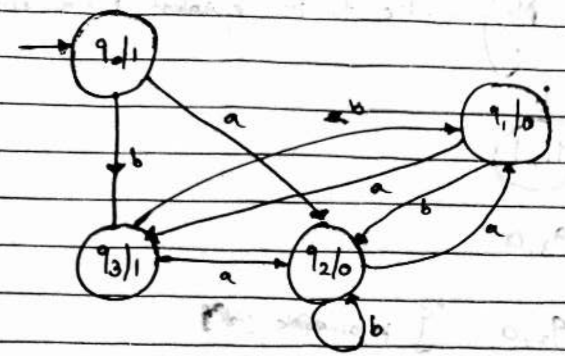
	a	b	o/p
q_0	q_1	q_3	1
q_1	q_3	q_1	0
q_2	q_0	q_3	0
q_3	q_3	q_2	1

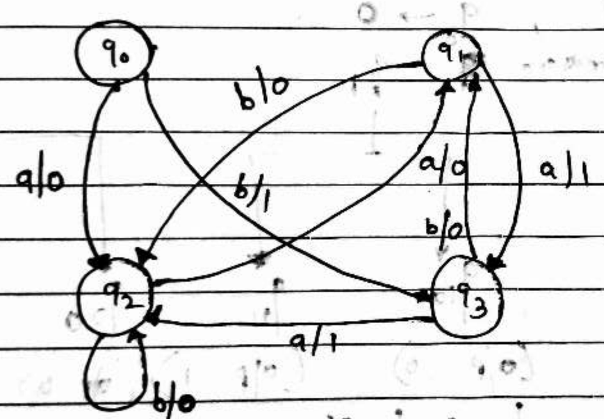
	a	o/p	b	o/p
q_0	q_1	0	q_3	1
q_1	q_3	1	q_1	0
q_2	q_0	1	q_3	1
q_3	q_3	1	q_2	0



machine description

Q.





Convert a given Mealy machine to its' equivalent Moore machine.

	a	b
→ q_0	$q_3, 0$	$q_1, 1$
q_1	$q_0, 1$	$q_3, 0$
q_2	$q_2, 1$	$q_2, 0$
q_3	$q_1, 0$	$q_0, 1$

} in accessible states

$q_0 \to 1$, $q_3 \to 0$

$q_1 \to$ breaks into

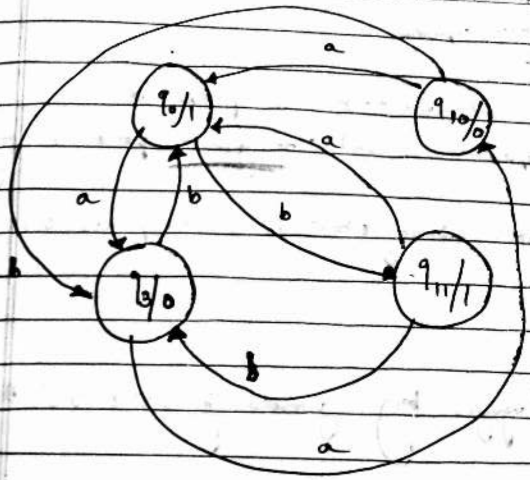
$q_1 \to q_{10}$ (o/p 0), q_{11} (o/p 1)

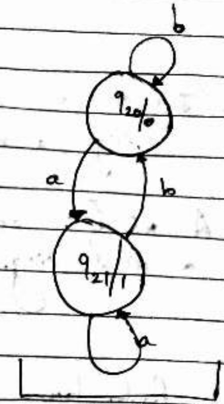
$q_2 \to q_{20}$ (o/p 0), q_{21} (o/p 1)

	a	b	o/p
q_0	q_3	q_{11}	1
q_{10}	q_0	q_3	0
q_{11}	q_0	q_3	1
q_{20}	q_{21}	q_{20}	0
q_{21}	q_{21}	q_{20}	1
q_3	q_{10}	q_0	0

} in accessible states

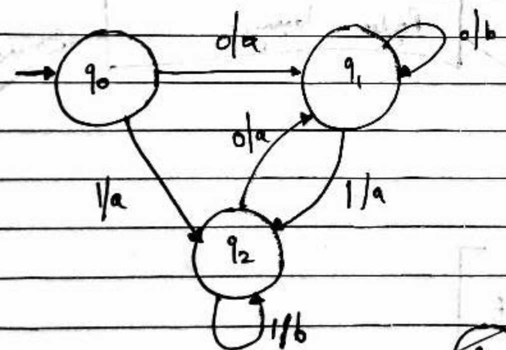
DATE / /





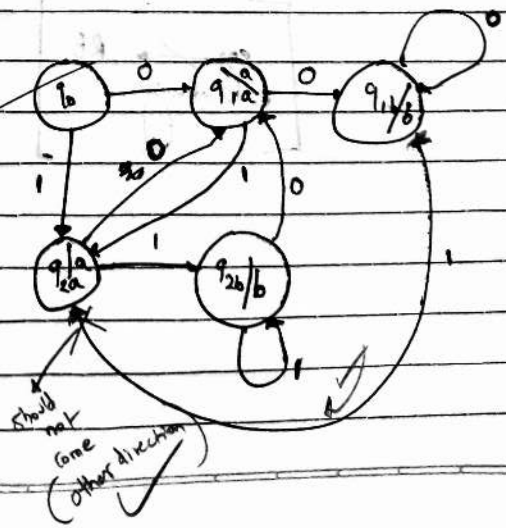
as it is in accessible state, we can remove it from the diagram.

Q. convert Mealy to Moore machine



$\Sigma = \{0, 1\}$

$O = \{a, b\}$



interesting

Maximum no. of states:- $m \times n$

DATE / / /

- FA
  - DFA
  - NFA
  - FA with o/p
    - moore
    - mealy

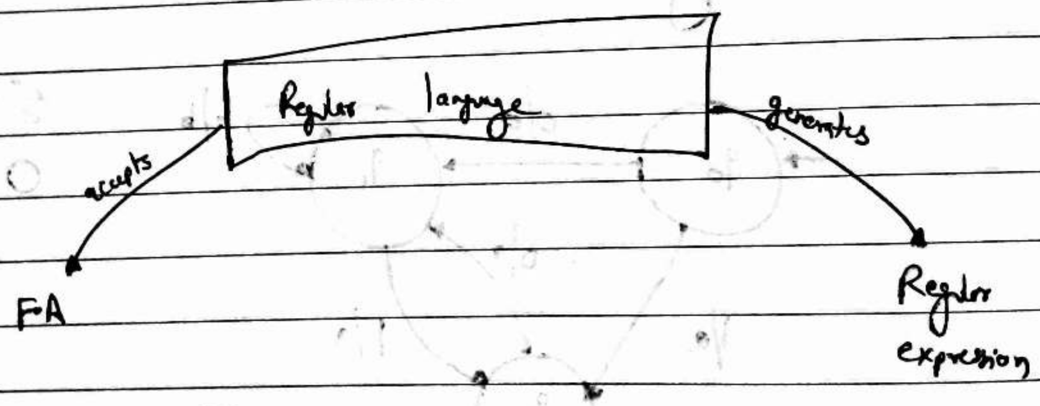
Relational expression

Just as finite automata are used to recognise pattern of strings, a rel^n expression is used to generate pattern of string.

A RE is an algebric formula whose value is a pattern consisting of a set of string.

Regular language → languages being generated by final states of DFA.

Regular language
$\downarrow$ generate
Regular expression



[~~eg~~ → RE
interesting]

Non primitive Operations in RE

1. union $(a+b) = \{a, b\}$
2. concatenation $(a \cdot b) = a \cdot b$ $(\neq b \cdot a)$
3. kleene closure $(a+b)^* = \{\epsilon, a, b, aa, ab, ba, bb, bbb, \ldots\}$

All the combinations

interesting length = 0

4. positive/nee-closure $(a+b)^+ = \{a, b, aa, bbb \ldots\}$ No ϵ

Precedence operation

1. ( )
2. * closure
3. + closure (positive closure)
4. · closure (concatenation)
5. +

$\phi = \{\ \}$ → Contains NOTHING

Primitive R.E

1. $\phi = \{\ \}$
2. $\epsilon / \lambda \rightarrow \{\{\epsilon\}\}$
3. $a \rightarrow \Sigma \rightarrow \{\{a\}, \{b\}, \{c\}, \ldots\}$

Explain given below R.E in terms of set:-

(a)

$r = a+b+c \longrightarrow L(r) = \{a, b, c\}$

$r = (a \cdot b + a) b \longrightarrow L(r) = \{abb, ab\}$

$r = (a + b \cdot a) \cdot a^* \longrightarrow L(r) = \{a \cdot a^*, b \cdot a \cdot a^*\}$

$= \{a, ba, aa, baa, aaa, baaa \ldots\}$

a or b having atleast 1 a at the end

DATE / /

$\Sigma = \{a, b\}$

write a regular expression for set of string ending with b.

$(a+b)^* \cdot b$

$(\Sigma)^* \cdot b$

$L(r) = \{b, ab, aab, abbab, \ldots\}$

$\downarrow$

$(a+b)^* \cdot b$

starts with a and ends with b

$a \cdot (a+b)^* \cdot b$

check

all the string containing exactly two consecutive

$\Sigma = \{0, 1\}$

$\{001\}^*$

~~wrong $(00$~~

$\{00+1\}^*$

$\{00\}^* \cdot \{100\}^* \cdot 1^*\}^*$

$\{100\}^* \cdot \{01\}$

$00 \cdot \{0+1\}^* + \{0+1+00\}^* + \{0+1\}^* \cdot 00$

interesting all the strings containing exactly two zeroes

$\Sigma = \{0, 1\}$

$\{0+1\}^* 0 \{0+1\}^* 0 \{0+1\}^*$

NOAH $\{00+1\}^*$

$1^* (01^*0 + 001^* + 1^*00) 1^*$

101^*

1010

$1^*01^*01^*$

$\Sigma = \{a, b\}$

① ~~regexp.~~ whose length is exactly ~~18~~ two

$L = \{aa + ab + ba + bb\}$

$\downarrow$

$(a+b)(a+b)$

② length exactly 3

$(a+b)(a+b)(a+b)$

③ whose length is atleast 2

$(a+b)(a+b)(a+b)^*$

OR

$(a+b)(a+b)^+$

④ At most 2

$\cancel{(a+b)(a+b)}$

$(a+b+\epsilon) \cdot (a+b+\epsilon)$ OR $\epsilon + (a+b)(a+b) + a+b$

④ even length string (always)

$\epsilon + (aa + ab + ba + bb)^*$

OR

$\big((a+b)(a+b)\big)^*$

a,b

$\downarrow$

aa, ab, ba, bb

......

$(a+b) \cdot \big((a+b)(a+b)\big)^*$

⑤ odd length string (always)

$(a+b) \cdot \big((a+b)(a+b)\big)^*$

$b^*(ab^*ab^*)^* ab^*$ $\quad a.(b^*.ab^*-a.b^*)^* + b^*$

aaaaba

⑥ length is divisible by 3

$((a+b)(a+b)(a+b))^+$

⑦ is behind

⑧ no. of a's exactly 2, b can be anything

$b^*.(ab^*a + aab^* + b^*aa).b^*$

OR

$(b^*ab^*).(b^*ab^*)$

redundant

OR

$\{b^*ab^*ab^*\}$

⑨ no of a's atleast 2

⑩ no. of a's atmost 2

⑪ no. of a's are even

⑫ no. of a's are odd

⑬ Starting and ending with different symbols

⑭ Starting and ending with same symbol.

⑮ String not ending with 01 $\quad \Sigma = \{0,1\}$

interesting

⑨ $(a+b)^*.(ab^*a + aab^* + b^*aa).(a+b)^*$

⑨ $a^*b^*ab^*ab^*a^*$

⑩ $b^*ab^*ab^* + b^*ab^* + b^*$

⑪ $((ab^*a)^* + (b^*aa)^* + (aab^*)^*)^*$

interesting

$(ab^*a)^* + (b^*aa)^* + (aab)^*$

OR

$(b^*ab^*ab^*)^*$

PAGE NO.
DATE: / /

aaaaba aaabaa

(12.) $(b^* \cdot a \cdot b^*)$

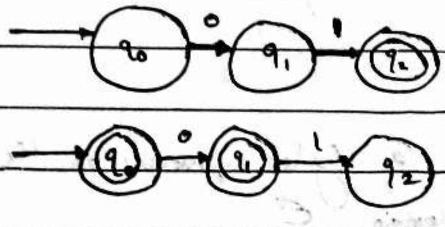
$a \cdot (b^* \cdot a \cdot b^* \cdot a \cdot b^*)^* + b^* + \epsilon$

$$\boxed{(b^* \cdot a \cdot b^*) \cdot (b^* \cdot a \cdot b^* \cdot a \cdot b^*)^* + b^* + \epsilon}$$

(13.) $a(a+b)^* b \;+\; b(a+b)^* a$

(14) $a(a+b)^* a \;+\; b(a+b)^* b$

(15.)



ending with '01'

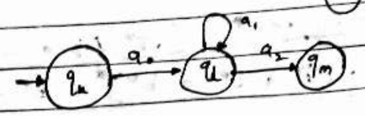
not ending with '01'

$$(0+1)^* (00 + 10 + 11) + \epsilon + 0 + 1$$

Arden's theorem

FA $\longrightarrow$ R.E

① Assume that we have FA consisting of M state



$q_L = \epsilon$

$q_2 = q_L \cdot a_0 + q_2 \cdot a_1$

$q_m = q_2 \cdot a_2$

② For each of state, identify the state eqⁿ.

③ Solve all m eqⁿs by substitute process using Arden's theorem.

④ Find RE of the final state solving Arden's theorem.

Arden's theorem :–

let P, Q, R be a regular expression on the set.

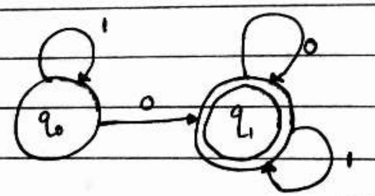
Then if P does not contain ϵ,

then eqⁿ $R = Q + RP$

has a unique solⁿ given by :–

$$R = Q \cdot P^*$$

Q.



$(1^* 0 (0+1)^*)$

interesting

$q_0 = q_0 \cdot 1 + \epsilon$ — ①

$q_1 = q_0 \cdot 0 + q_1 \cdot 0 + q_1 \cdot 1$

$= q_0 \cdot 0 + q_1 (0+1)$ — ②

[illegible]

$R = Q + RP = QP^*$

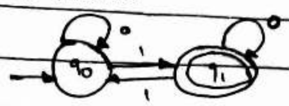
PAGE No. DATE / /

$q_1 = q_0 \cdot 0 \cdot (0+1)^*$
where $q_0 = \epsilon \cdot 1^* = 1^*$

$q_1 = 1^* \cdot 0 \cdot (0+1)^*$

$R = Q + RP$
$= QP^*$

using arden's theorem



$q_0 = \epsilon + q_0 \cdot 0 + q_1 \cdot 1$
$q_1 = q_1 \cdot 0 + q_0 \cdot 1$
$R = Q + RP$
$= (q_0 \cdot 1) + 0^*$

$q_0 = q_0 \cdot 0 + \epsilon$, $\quad q_0 = \epsilon \cdot 0^*$
$q_1 = q_0 \cdot 1 + q_1 \cdot 0$

$q_0 = \underbrace{q_0 \cdot 0}_{RP} + \underbrace{q_1 \cdot 1 + \epsilon}_{Q}$

$q_1 = \underbrace{0^* \cdot 1}_{Q} + \underbrace{q_1 \cdot 0}_{RP}$

$q_1 = \underbrace{q_1 \cdot 0}_{RP} + \underbrace{q_0 \cdot 1}_{Q}$

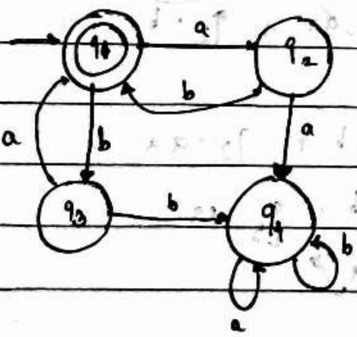
$q_1 = 0^* 1 \cdot 0^*$

$R = Q + RP$
$= QP^*$

$q_0 = (q_1 \cdot 1 + \epsilon) \cdot 0^*$

$q_1 = (q_0 \cdot 1) \cdot 0^*$

Q.



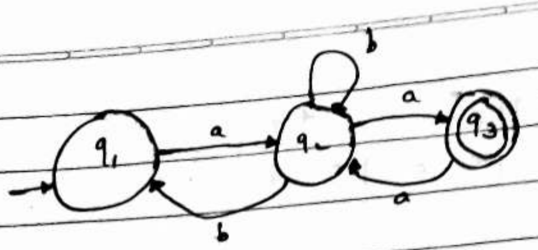
$(ab+ba)^*$

$q_1 = \epsilon + q_3 \cdot a + q_2 \cdot b \quad$ — (1)
$q_2 = a \cdot q_1 \quad$ — (2)
$q_3 = b \cdot q_1 \quad$ — (3)
$q_4 = a \cdot q_2 + b \cdot q_3 + q_4 \cdot a + q_4 \cdot b \quad$ — (4)

put (2), (3) in (1)

$q_1 = \epsilon + b \cdot a \cdot q_1 + a \cdot b \cdot q_1$
$R = Q + RP = QP^*$
$= \epsilon \cdot (b \cdot a + ab)^* = (ab+ba)^*$

COLLEGE BOOK



$\cancel{(a+b)^* \cdot aa}$

$q_1 = \epsilon + q_2 \cdot b$

aa

$q_2 = q_1 \cdot a + q_2 \cdot b + q_3 \cdot a$

$q_3 = q_2 \cdot a$

$\cancel{q_1 \cdot a + q_3 \cdot a = a + q_2 \cdot b \cdot a + q_2 \cdot ab}$

$\cancel{q_2 = a + q_2 \cdot b \cdot a + q_2 \cdot a \cdot b + q_2 \cdot b}$

$\cancel{R = Q + RP}$

$\cancel{q_2 = a \cdot (b \cdot a + ab + b)^*}$

$\cancel{q_1 \cdot a + q_3 \cdot a = a + q_2 \cdot b \cdot a + q_2 \cdot aa}$

$\cancel{q_2 = a + q_2 \cdot b \cdot a + q_2 \cdot aa + q_2 \cdot b}$

$q_2 = (\epsilon + q_2 \cdot b)a + q_2 \cdot b + q_2 \cdot aa$

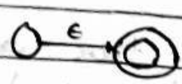
$= a + q_2 \cdot b \cdot a + q_2 \cdot b + q_2 \cdot aa$

$= a \cdot (b \cdot a + b + aa)^*$

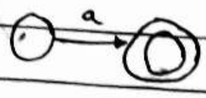
$\cancel{q_2 = a + q_2 ba + q_2 b + a}$

Regular expression to Finite machine !

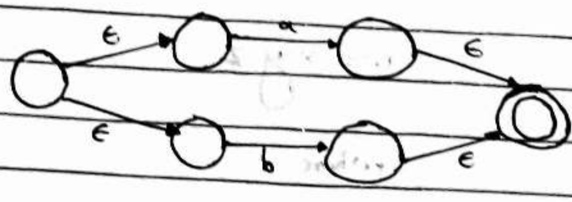
① ϵ



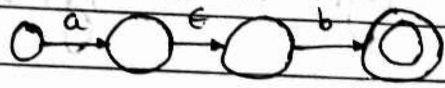
② a



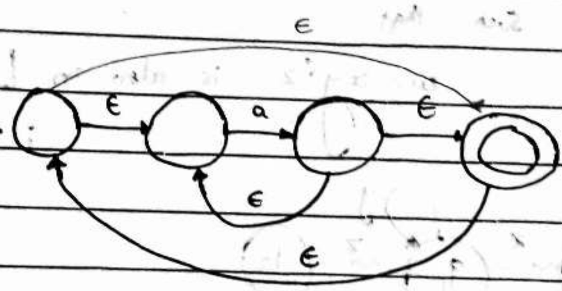
③ a+b



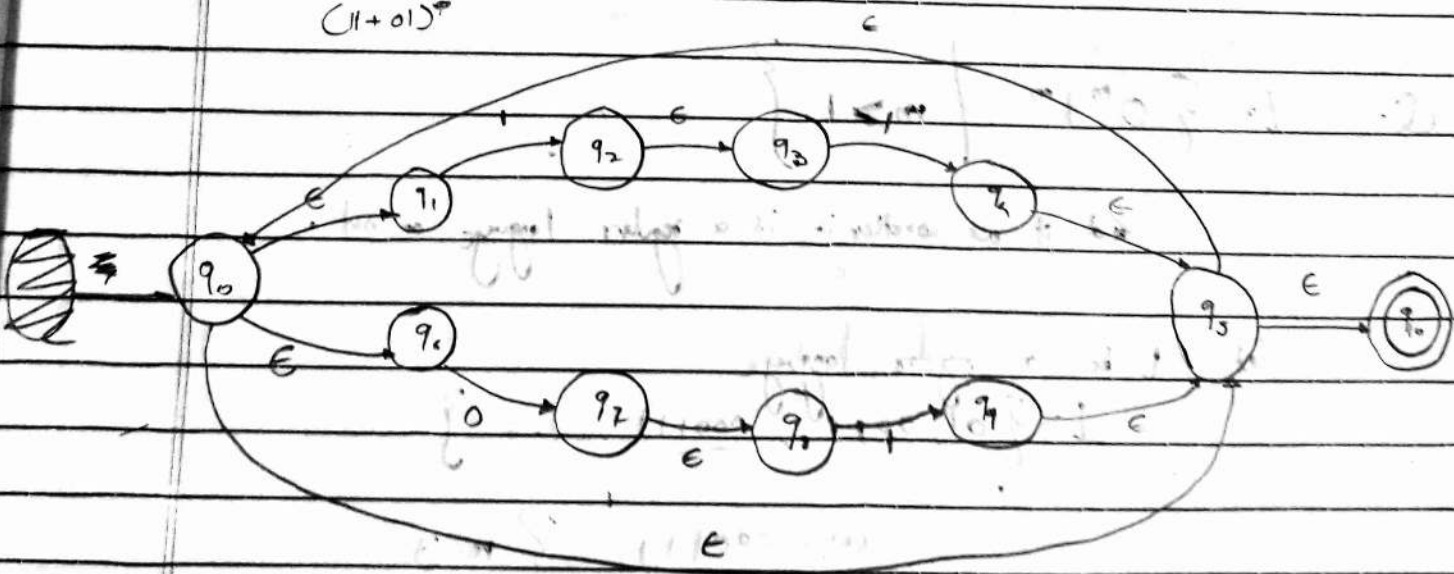
④ a·b



⑤ a^*



$(11+01)^*$



$\epsilon\text{-cl.}(q_0) = \{q_0, q_1, q_6, q_0, q_{10}\}$

$\epsilon\text{-cl.}(q_1) = \{q_1\}$

$\epsilon\text{-cl.}(q_2) = \{q_2, q_3\}$

$\epsilon\text{-cl.}(q_3) = \{q_5, q_0, q_1, q_2\}$

$\epsilon\text{-cl.}(q_4) = \{q_4, q_5, q_0, q_{10}\}$

very interesting → is a negative test (it will tell No) (we assume at beginning, it is a regular language)

Pumping Lemma

(prove that some languages are not regular)

interesting

$|w| \geq m$
$w = xyz$
$|xy| \leq m$
$|y| \geq 1$
then
xy^iz is also in L for all values of $i = 0, 1, 2, 3 \ldots$

Let L be an infinite regular language, then there exists some +ve integer 'm' such that any $w \in L$ with length of $|w| \geq m$ — (1)

Can be decomposed as

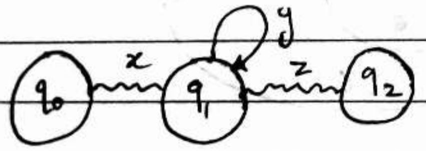
$$w = xyz$$

where

$$|xy| \leq m \text{ and } |y| \geq 1 \quad (2)$$

such that

$$w = xy^iz \text{ is also in } L \text{ for all } i = 0, 1, 2 \ldots$$



Q. $L = \{0^m 1^m \mid m \geq 1\}$

find it out whether it is a regular language or not.

let L be a regular language

$L = \{01, 0011, \underline{000111} \ldots\}$

$w = 000111$
$w = 0^3 1^3$
$|w| = 6$

$m = 3$
(if we take $m = 3$ we get $w = 000111$)

If non regular languages have memory, they will be able to solve the problems State machine.

PAGE NO.	
DATE	/ /

$6 \geq 3$ — 1 satisfied

$w = \underbrace{00}_{x}\underbrace{0}_{y}\underbrace{111}_{z}$

$|xy| \leq m$ and $|y| \geq 1$

$3 \leq 3$ $\quad$ $2 \geq 1$ — 2 satisfied

$w = xy^iz$

$i=0$, $w = 0111 \notin L$ (Not regular language)

not belong to L

Q. $L = \{ww \mid w \in \{0,1\}^*\}$ $\quad$ m=?

$L = \{ \epsilon, 00, 11, 0101, 1010, 10001000, 100100 \ldots \}$

$\cancel{w = 100100}$ ✓ | one example is ~~not~~ suited
$w = 1^m 0^{2n} 1^m 0^{2n}$ | we can take this string

$m=1$ $m+2$
$1\ 00$

$w = 1100100$ | $m=2$
$= 1^2 0^2 1^2 0^2$
$|w| = 8$

$8 \geq 2$ — 1 satisfied

$w = \underbrace{1}_{x}\underbrace{1}_{y}\underbrace{00100}_{z}$

$|xy| \leq m$, $\quad$ $|y| \geq 1$

$2 \leq 2$ $\quad$ $1 \geq 1$ — 2 satisfied

$w = xy^iz \xrightarrow{i=0} 1001100 \notin L$ (Not regular language)

- Pumping lemma is used to prove for irregularity of the language
- If there exists atleast 1 string made from the language and pumping lemma then prove that it is not in L, then L is surely not a regular language.

Q. $L = \{a^p \mid p \text{ is a prime no.}\}$

$L = \{w w^R \mid w \in \{0,1\}^*\}$ $w = 01010101$

$|xy| \le 3$

xy^iz

$0\,(10)\,0$ $i=0$

$0\,0\,1\,0\,1\,0\,1$

$L = \{a^n b^{n+1} \mid \Sigma = \{a,b\}^+\}$

$L = \{a^p\}$

$= \{a^2, a^3, a^5, a^7 \ldots\}$

$= \{aa, aaa, aaaaa, aaaaaaa \ldots\}$

$w = aaaaa$ $m = 5$

$= a^5$

$|w| \ge m$

$5 \ge 5$ — (1)

$w = \underset{x}{a}\,\underset{y}{a}\,\underset{z}{aaa}$

$|xy| \le m$ $|y| \ge 1$

$2 \le 5$

$xy^iz \rightarrow aa^iaaa$

$i=1 \rightarrow aaaaa$

$i=0 \rightarrow aaaa$ X

(Not regular)

$L = \{ a^n b^{n+1} \mid \Sigma = \{a,b\}^+ \}$

$L = \{ abb, aabbb, aaabbbb, b, \ldots \}$

$a^m b^{m+1}$

$m = 3$

$|w| = \{ aaabbbb \}$

$|w| = 7$

For m,

we have to take a relationship which satisfies $\lambda \in L$ for that m.

eg. $100100 \rightarrow 1^m 0^{m+2}$

$aaabbbb \rightarrow a^n b^{n+1}$

$aaaaa \rightarrow a^n$

$|w| \geq m$

$7 \geq 3$

$w = \underbrace{a}_{x}\underbrace{a}_{y}\underbrace{abbbb}_{z}$

$|xy| \leq m$ $\quad$ $|y| \geq 1$

$2 \leq 3$ $\quad$ $1 \geq 1$

$xy^iz \rightarrow aa^iabbbb$

$i = 0$ ✗

(Not regular)

Grammars

Grammars will generate $\boxed{\text{valid}}$ strings as for some languages

Creating R.E is not easy.

for english language

alphabet (26)
↓
words $\xrightarrow[\text{98}]{\text{finite}}$ dict
↓
sentences $\xrightarrow{\text{rules}}$ grammar

For every language there exists acceptor like FSM and there also exists generator like Grammar

Generator does not check for any string that is ~~excep~~ whether it is accepted or not, but it generates the entire language.

eg. $S \to aS \mid bS \mid \epsilon$ → This $\boxed{\text{production rule}}$

mapped with ~~[illegible]~~ $\xrightarrow{\text{R.E}}$ $(a+b)^*$

where '|' denotes OR

$$\left.\begin{array}{l} S \to aS \\ S \to bS \\ S \to \epsilon \end{array}\right\}$$

will generate the string with any number of a and b.

$\{a,b\} \to$ Terminal

$\Sigma = \{a,b\}$

$\{S\} \to$ variable

$L = \{a, b, \epsilon, ab, baa, \ldots\}$

$aS \to abS \to ab\epsilon \to ab$

$aS \to a\epsilon \to a$

$bS \to b(aS) \to baaS \to baa\epsilon \to baa$

Q. $S \to aA$

$A \to aA \mid bA \mid \epsilon$ $\xrightarrow[\text{to R.E}]{\text{mapped}}$ $a(a+b)^*$

$\Sigma = \{a, b\}$

Q. $S \to aSb$

$S \to ab$ $\qquad a^n b^n$ ✓

$a^+ b^+$ (also correct)

Q. generate a grammar for a set of string of length 2.

$\Sigma = \{a, b\}$ (Terminal set)

$L = \{aa, ab, ba, bb\}$

Production rule:

$S \to AA$

$A \to a \mid b$

(where A is Non-terminal)

Q. generate a grammar for language

$a^n \mid n \geq 0$

$S \to A$

$A \to aA \mid \epsilon$

Q. $(a+b)^*$

$S \to aS \mid bS \mid \epsilon$

Q. String of length atleast 2, $L = \{a, b\}$

$S \to AA$
$A \to a \mid aA \mid b \mid bA$

Q. atmost 2
$L = \{a, b\}$

$S \to AA$
$A \to a \mid b \mid \epsilon$

Q. String starts with a, ends with b

$S \to aAb$
$A \to aA \mid bA \mid \epsilon$

Q. starts and ends with different symbols

$S \to aAb \mid bAa$
$A \to aA \mid bA \mid \epsilon$

Q. $a^n b^n \mid n \geq 1$

$S \to aAb$
$A \to aAb \mid \epsilon$

DATE / /

Chomsky Hierarchy

Grammar

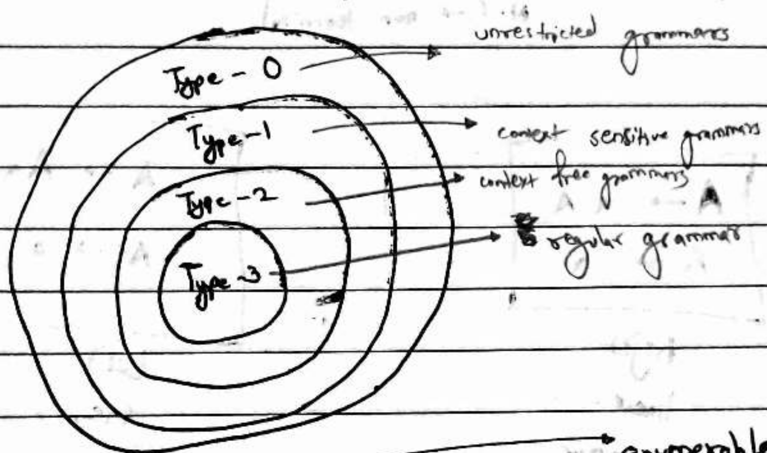
$G = (V, T, P, S)$

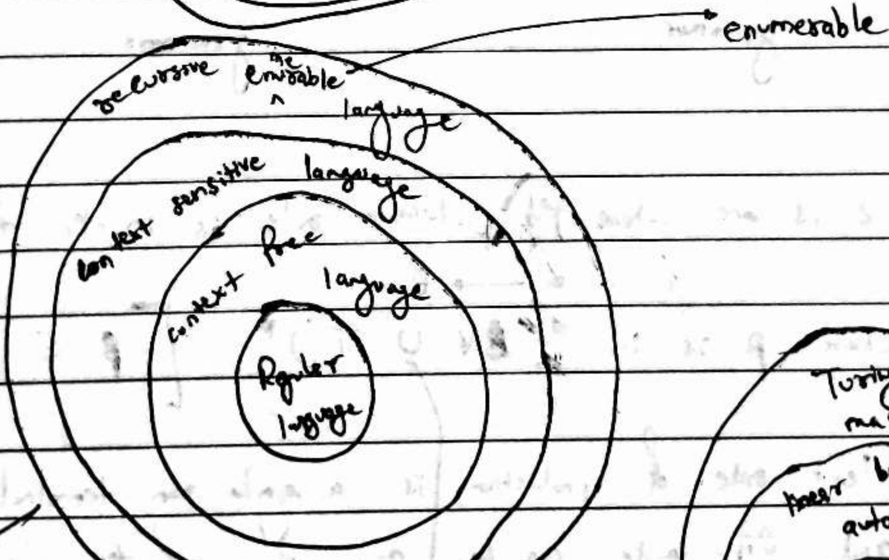
V → non terminals

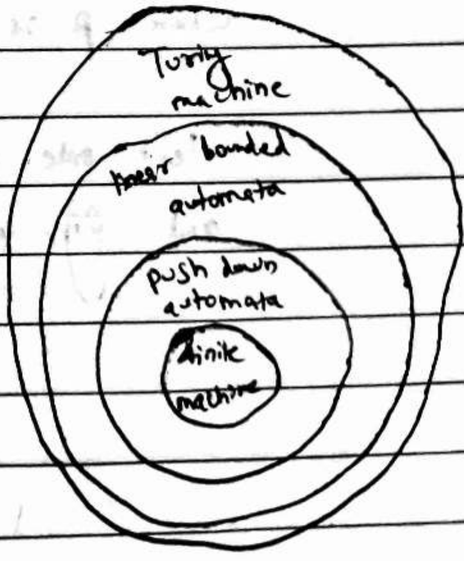
T → terminals

P → Rules of production

S → Starting







Seen

11/9/1

$V \rightarrow$ non terminal

$T \rightarrow$ terminal

Type 3 grammar is one where every production rule is in the form of ~~non terminal~~ goes to terminal or

$NT \rightarrow (T)(N \cdot T)$ or

$N \cdot T \rightarrow (T)$ or

$N \cdot T \rightarrow (N \cdot T) \cdot (T)$

$T \rightarrow$ terminal

$N \cdot T \rightarrow$ non terminal

$$\boxed{\begin{array}{l} A \rightarrow aA \\ A \rightarrow a \end{array}}$$

Right linear grammar

$$\boxed{\begin{array}{l} A \rightarrow Aa \\ A \rightarrow a \end{array}}$$

Left linear grammar

Type 2 is one where the production rule is in the form of:-

$\alpha \longrightarrow B$

where β is in $(V \cup T)^*$ $[\beta \in (V \cup T)^*]$

(union)

left side of production is a single non terminal or variable and right side can be any string of terminals and nonterminals.

* Type 1 grammar is one whose production rule is of

$$\alpha \rightarrow \beta$$

whose α and β are in $(V \cup T)^*$

i.e.

① $\alpha, \beta \in (V \cup T)^*$

and ② $\alpha \notin \epsilon$

and ③ $|\beta| \geq |\alpha|$ → width alpha

(length of β is $\geq$ length of alpha)

Type 0 grammar is one whose every production rule is of form :-

$$\alpha \rightarrow \beta$$

where $\alpha, \beta \in (V \cup T)^*$

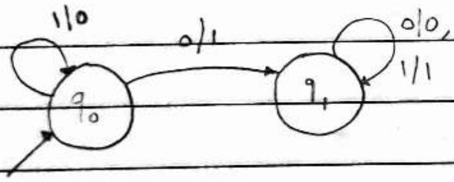
where α, β can be any string terminal or non terminal.

→ there must be atleast 1 variable on the left side of production.

It is capable of satisfying the largest number of language.

1010

mealy machine to increment the val. of binary no. by 1



	0	o/p	1	o/p
q_0	q_1	1	q_0	0
q_1	q_1	0	q_1	1

8 4 2 1

11011 → 27

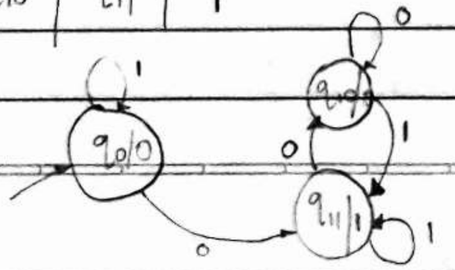
11100 → 28

10

11

	0	1	o/p
q_0	q_{10}	q_0	0
q_{10}	q_{10}	q_{11}	0
q_{11}	q_{10}	q_{11}	1

	0	1
q_0	q_1, 1	q_0, 0
q_1	q_1, 0	q_1, 1



minimize the DFA :-

States	0	1
→ q_0	q_1	q_4
q_1	q_2	q_5
* q_2	q_3	q_7
q_3	q_4	q_7
q_4	q_5	q_8
* q_5	q_6	q_1
q_6	q_7	q_1
q_7	q_8	q_2
* q_8	q_0	q_4

$$P_0 = \overset{A}{\{q_0, q_1, q_3, q_4, q_6, q_7\}} \quad \overset{B}{\{q_2, q_5, q_8\}}$$

$\delta(q_0, 0) = A$
$\delta(q_0, 1) = A$

$\delta(q_1, 0) = B$
$\delta(q_1, 1) = B$

$\delta(q_2, 0) = A$
$\delta(q_2, 1) = A$

$\delta(q_3, 0) = A$
$\delta(q_3, 1) = A$

$\delta(q_4, 0) = B$
$\delta(q_4, 1) = B$

$\delta(q_5, 0) = A$
$\delta(q_5, 1) = A$

$\delta(q_6, 0) = A$
$\delta(q_6, 1) = A$

$\delta(q_7, 0) = B$
$\delta(q_7, 1) = B$

$\delta(q_8, 0) = A$
$\delta(q_8, 1) = A$

Final states

$$P_1 = \overset{A}{\{q_1, q_4, q_7\}} \quad \overset{B}{\{q_2, q_5, q_8\}} \quad \overset{C}{\{q_0, q_3, q_6\}}$$

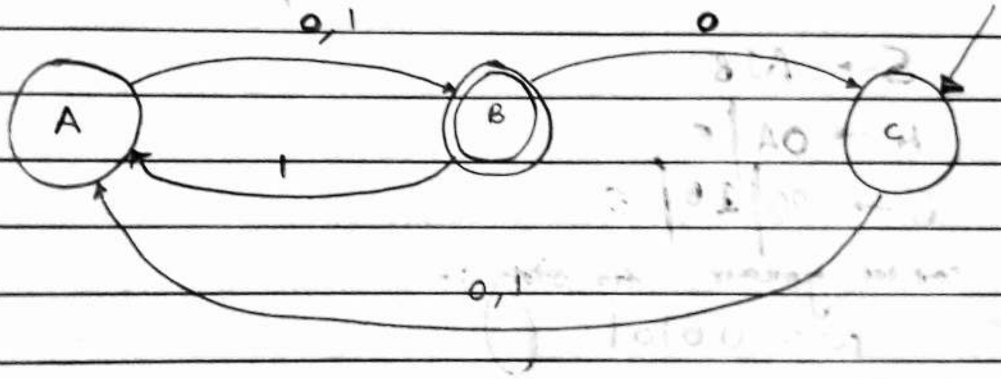
$q_1 \xrightarrow{0} B$, $q_1 \xrightarrow{1} B$ $\quad$ $q_0 \xrightarrow{0} A$, $q_0 \xrightarrow{1} A$ $\quad$ $q_2 \xrightarrow{0} C$, $q_2 \xrightarrow{1} A$ F $\quad$ $q_3 \xrightarrow{0} A$, $q_3 \xrightarrow{1} A$

$q_4 \xrightarrow{0} B$, $q_4 \xrightarrow{1} B$ $\quad$ $q_5 \xrightarrow{0} C$, $q_5 \xrightarrow{1} A$ F $\quad$ $q_6 \xrightarrow{0} A$, $q_6 \xrightarrow{1} A$ $\quad$ $q_7 \xrightarrow{0} B$, $q_7 \xrightarrow{1} B$

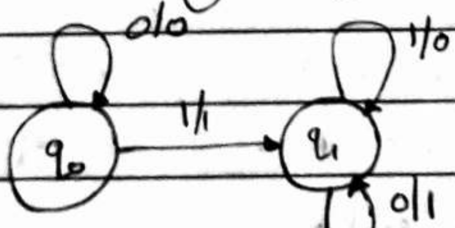
$q_8 \xrightarrow{0} C$, $q_8 \xrightarrow{1} A$ F

A $\qquad$ C $\qquad$ B

$P_2 = \{q_1, q_4, q_7\} \quad \{q_0, q_3, q_6\} \quad \{q_2, q_5, q_8\}$



2's complement using mealy machine



DATE / / /

Context free grammar

$G = (V, T, P, S)$

V → variables

T → terminals

P → production rule

S → start symbol

Grammar

- sentential form
- parse tree

Q. $S \rightarrow A1B$

$A \rightarrow 0A \mid \epsilon$

$B \rightarrow 0B \mid 1B \mid \epsilon$

consider grammar for string :—

$w = 00101$

find the following :—

1. left most derivation
2. right most derivation
3. parse tree

(wrt non-terminating symbol)

→ left most derivation → left most non-terminating symbol first

S → start symbol

$T = \{0, 1\}$

$V = \{0, A, B\}$

PAGE No.
DATE / /

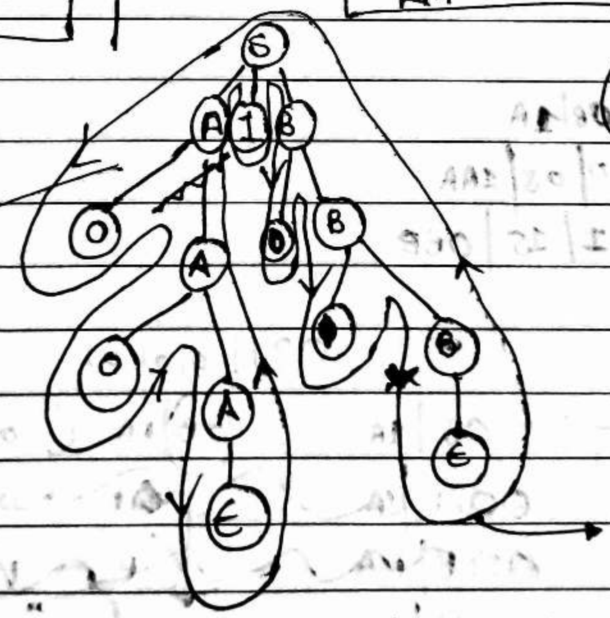
Left most derivation	Right most derivation
$S \to A1B$	$S \to A1B$
$S \to (A \to 0A)$	$S \to (B \to 0B)$
$0A1B$	$A10B$
$(A \to 0A)$	$(B \to 1B)$
$00A1B$	$A101B$
$(A \to \epsilon)$	$(B \to \epsilon)$
$001B$	$A101$
$(B \to 0B)$	$(A \to 0A)$
$0010B$	$0A101$
$(B \to 1B)$	$(A \to 0A)$
$00101B$	$00A101$
$(B \to \epsilon)$	$(A \to \epsilon)$
$S \to 00101$	$S \to 00101$

interesting representation

$\boxed{S \overset{*}{\underset{LM}{\Rightarrow}} 00101}$ $\boxed{S \overset{*}{\underset{RMT}{\Rightarrow}} 00101}$

Sentential representation of grammar

Parse tree:- (option to choose rightmost or left most)



left most derivation

$\rightarrow$ 00101

{ derivation starts from start symbols and final string should consist of terminals.

PAGE NO.
DATE / /

• Derivations represented

sentential form

① derivation from start symbol produce a string by applying finite no. of production rules.

② Rep:– $S \overset{*}{\Longrightarrow} \alpha$
RM/LM

parse tree

① it is ordered rooted tree and the graphical representation of how the sentence is derived from the start symbol given

② Root must be labeled by start symbol

③ Vertex labeled by non terminal

④ leaves labeled by terminal or NULL.

Q. S → 0B|1A
A → 0|0S|1AA
B → 1|1S|0BB

(w = 0011 01010) left most derivation

LMD:– 0B | 1A 0B|1A (0B → 0BB)
~~00BB|1A~~ (0B → 0BB)
~~001B|1A~~
~~0011|1A~~
0 0 BB (0B → 0BB)
0 0 1S B (B → 1S)
0011AB (S → 1A)
00110SB (A → 0S)
001101AB (S → 1A)
0011010SB (A → 0S)
00110101AB (S → 1A)
001101010B (A → 0) → 0011010101

0OBB
001S1S
0011A1A
00110S10S
00110 1A101A
0011010 1010

0B10S
0B

right most derivation → if in previous Q, we solve right most first derivation is found

OB | IA

OBAh

0B105 Q. $S \to S+S \mid S*S \mid S \mid a$

OB

$\underline{a + a * a}$

$V = \{S\}$

$T = \{a, *, +\}$

$S+S$ $(S \to a)$

↓

$a + S$ $(S \to S*S)$

$a + S*S$ $(S \to a)$

$a + a*S$ $(S \to a)$

$a + a*a$

OR

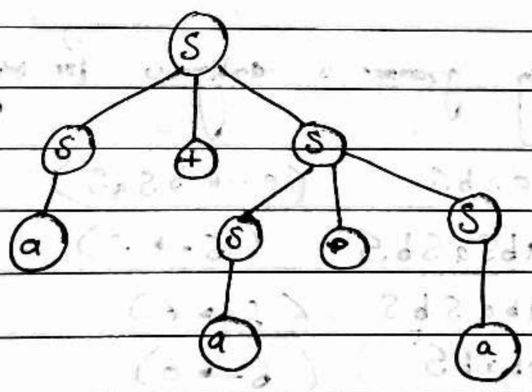
$S \to S*S$

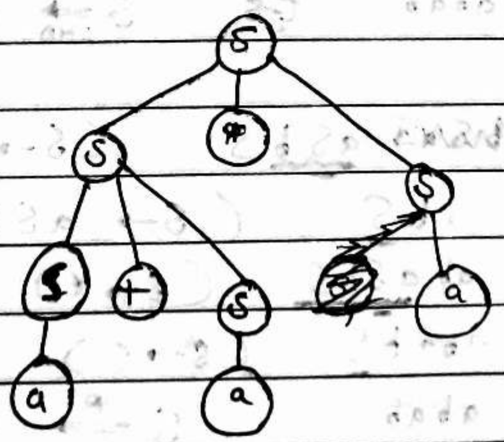
$S \to S+S*S$

$S \to a+S*S$

$S \to a+a*S$

$S \to a+a*a$





This is ambiguous grammar.

(more than 1 left most derivation or more than 1 right most derivation.)

DATE / /

Ambiguous Grammar

more than one LMD

or

more than one RMD

or more than one parse tree

A grammar uniquely determines a structure for each string in its language, but not each grammar can produce strings uniquely.

A grammar is said to be ambiguous if there exists atleast 1 string, which can be generated in more than 1 way, following LMD or RMD.

Some strings can have two different parse trees.

Q. $S \rightarrow aSbS$

$S \rightarrow bSaS$

$S \rightarrow \epsilon$

show that following grammar is ambiguous for string 'abab'

LMD $S \rightarrow a\underline{S}bS$ $(S \rightarrow bSaS)$

$S \rightarrow ab\underline{S}aSbS$ $(S \rightarrow \epsilon)$

$S \rightarrow abaSbS$ $(S \rightarrow \epsilon)$

$S \rightarrow ababS$ $(S \rightarrow \epsilon)$

$S \rightarrow abab$ $S \xRightarrow[LMD]{*} abab$

LMD $S \rightarrow \cancel{bSaS}$ $a\underline{S}bS$ $(S \rightarrow \epsilon)$

$S \rightarrow ab\underline{S}$ $(S \rightarrow aSbS)$

$S \rightarrow ab\,a\underline{S}bS$ $(S \rightarrow \epsilon)$

$S \rightarrow abab\underline{S}$ $(S \rightarrow \epsilon)$

$S \rightarrow abab$

$S \xRightarrow[LMD]{*} abab$

$(0+1)^*(01)+\epsilon$

ictS → ibtS → ibtictSeS → ibtibtSeS → ibtibtaea
ictSeS → ibtSeS → ibtictSeS → ibtibtSeS → ibtibtaeS → ibtibtaea

Q. $S \rightarrow iCtS \mid iCtSeS \mid a$
$C \rightarrow b$

'ibtibtaea'

ictS
ict ictSeS
ictictSea
ictictaea
ictibtaea
ibtibtaea

- LMD
- RMD
- parse tree
- check if it is ambiguous

Simplification of Grammar

~~interesting~~ This simplication can be applied on any type of grammar (not necessarily context free)

CFG → CNF, GNF

- A grammar written in a simple form is easy to analyze. Certain restrictions are imposed on simplified grammar.
- Simplification of grammar involves transforming grammar into equivalent form that satisfies certain restrictions on its form.

A grammar can be simplified by eliminating :—

① Useless symbols → non reachable, non generating
② Null (ϵ) production
③ Unit production

⑴ $S \rightarrow Aa \mid Bb \mid a \mid b$
$A \rightarrow Aa \mid a$
$B \rightarrow bB$

we remove the non reachable states / non generating state

$S \rightarrow Aa \mid a \mid b$
$A \rightarrow Aa \mid a$

rough work

$S \rightarrow Aa \mid Sb \mid a \mid b \mid bS$
$A \rightarrow Aa \mid a$

(2) $S \to A$
$A \to aA$
$A \to \epsilon$
$\boxed{B \to bA}$ non reachable state

Q. $S \to aSa \mid bSb \mid \epsilon$

$S \to \epsilon$

$S \to aSa$
$\to aa \quad (S \to \epsilon)$

$S \to bSb$
$\to bb \quad (S \to \epsilon)$

$$\boxed{S \to aSa \mid bSb \mid aa \mid bb}$$

Q. $S \to aS \mid bA$
$A \to aA \mid \epsilon$

$S \to aS \mid bA$
$A \to aA \mid a$

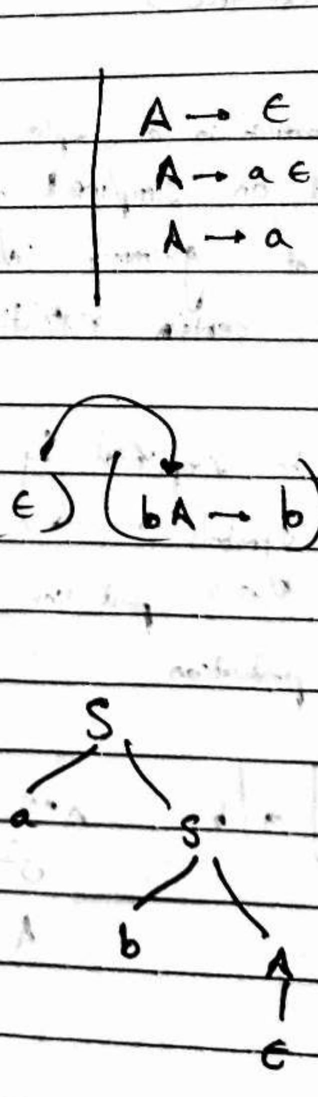
$A \to \epsilon$
$A \to a\epsilon$
$A \to a$

$S \to aS \mid bA$
$A \to aA \mid a$

$(b \to \epsilon) \quad (bA \to b)$

$S \to aS \mid bA \mid b$
$A \to aA \mid a$

$$\boxed{\begin{array}{l} S \to aS \mid bA \mid b \mid ab \\ A \to aA \mid a \end{array}}$$



PAGE NO.
DATE / /

Normal Form

- CNF (chomsky Normal form)
- GNF (greibach Normal form)

CNF (Chomsky Normal form)
Context free grammars can be written in standard form known as Normal Form.

These Normal Form impose certain restriction on productions of CFG.
These are 2 Normal forms:—

CNF and GNF

CNF :— A CFG without ϵ production is said to be in CNF if every production is in the form of $A \rightarrow BC$.

where

$$A, B, C \in V$$

$$A \rightarrow a \quad , a \in T$$

i.e. Rules $A \rightarrow BC$

where $A, B, C \in V$

$A \rightarrow a \quad , a \in T$

Algorithm for CFG to CNF:—

1. Grammar should be simplified
2. Every variable deriving a string of length 2 or more should consist only of variables.
3. Every non-CFG production deriving more than 2 variable can be removed of production, each deriving a string of 2 non terminals.

$S \to ABC \mid BaD$

eg:- $A \to PQRS$
$A \to PX$
$X \to QRS$
$X \to QY$
$Y \to RS$

$\longrightarrow$
$A \to PX$
$X \to QY$
$Y \to RS$

Q. $A \to PQaR$
$C_a \to a$
$A \to PQC_aR$
$A \to PX$
$X \to QC_aR$
$X \to QY$
$Y \to C_aR$

$\longrightarrow$
$A \to PX$
$X \to QY$
$Y \to C_aR$
$C_a \to a$

Q. $S \to aSa \mid bSb \mid a \mid b \mid aa \mid bb$

$G = (V, T, P, S)$
$V = \{S\}$
$T = \{a, b\}$
$S = \{S\}$

$S \to a$
$S \to b$
both are in C.N.F

$S \to aSa \mid bSb \mid aa \mid bb$

$S \to C_a S C_a$
$C_a \to a$
$S \to C_a A$
$A \to S C_a$

$S \to C_b S C_b$
$C_b \to b$
$S \to C_b B$
$B \to S C_b$

$S \to AA \mid a$
$A \to SS \mid b$

$S \to a$
$S \to AA$
$A \to SS$
$A \to b$

$S \to SSA \mid bA \mid a$
$A \to SS \mid b$

PAGE No.
DATE / /

$S \to aa \longrightarrow S \to C_a C_a$
$S \to bb \longrightarrow S \to C_b C_b$
Rule (2) (last page algorithm)

Q. $S \to ASA \mid aB$
$A \to B \mid S$
$B \to b \mid \epsilon$

when we remove $B \to \epsilon$

Removing ϵ

not needed

$S \to ASA \mid aB \mid a$
$A \to B \mid S \mid \epsilon$
$B \to b$

$A \to \epsilon$ removed

$S \to ASA \mid aB \mid a \mid SA \mid AS \mid S$
$A \to B \mid S$
$B \to b$

$S \to S$, $A \to B$, $A \to S$

Removing unit production

$S \to 01A$
$A \to 0S1S \mid A \mid \epsilon$

$B \to b$
$A \to b \mid ASA \mid aB \mid a \mid SA \mid AS \mid S$
$S \to ASA \mid aB \mid a \mid SA \mid AS$

$A \to AA \mid B$

$(A \to S)$ (unit productions)

$A \to BA'$
$A' \to A\alpha A' \mid \epsilon$

$B \to b$
$A \to b \mid ASA \mid aB \mid a \mid SA \mid AS$
$S \to ASA \mid aB \mid a \mid SA \mid AS$

$S \to bAS' \mid aS'$
$S' \to SAS' \mid \epsilon$
$A \to SS \mid b$

Convert to CNF

$S \to bAS' \mid aS' \mid bA \mid a$
$S' \to SAS' \mid SA$
$A \to SS \mid b$

$\longrightarrow$

$S \to bAS' \mid aS' \mid bA \mid a$
$S' \to bAS'AS' \mid bAS'A \mid aS'AS' \mid aS'A \mid bAAS' \mid bAA \mid aAS' \mid aA$
$A \to SS \mid b$

$S \to G_P \mid G \mid PP \mid PQ \mid QP \mid PR \mid C_bQ \mid C_b$ $\quad S \to P \mid PP \mid PQ \mid QP \mid PQPR \mid Q$

$R \to QP$

$P \to GP \mid G$ $\quad P \to C_aP \mid C_a$ $\quad R \to QP$

$Q \to C_bQ \mid C_b$ $\quad Q \to C_bQ \mid C_b$

$C_a \to 0$ $\quad C_a = 0$

$C_b \to 1$ $\quad C_b = 1$

Q. $S \to PQP$ $\qquad S \to PQ \mid QP \mid Q \mid PQP$ $\qquad S \to P \mid PP \mid PQ \mid QP \mid PQP \mid Q$

$P \to 0P \mid \epsilon \to$ $\qquad P \to 0P \mid 0$ $\to$ $\qquad P \to 0P \mid 0$

$Q \to 1Q \mid \epsilon$ $\qquad Q \to 1Q \mid \epsilon$ $\qquad Q \to 1Q \mid 1$

Pushdown automata :- Can be viewed as finite automata with stack

An added stack provides memory

- push
- pop
- no operation

PDA can do :- (i) read input symbol

(ii) perform stack operation

(a) push (b) pop (c) check empty condition by initial stack symbol

(d) read top symbol of stack without pop (peek)

(iii) make state change

PDA $\delta \to (Q \times \Sigma \cup \{\epsilon\} \times \gamma) \to (Q, \gamma^*)$

$\delta \to (Q \times \gamma) \to (Q, \gamma, (L, R))$ Turing machine

$L = a^n b^n \mid n \geq 1$

$L = \{ab, aabb, aaabbb, \ldots\}$

Three types of transition behaviour.

push:- $\delta(q_0, a, z_0) - (q_0, az_0)$

pop:- $\delta(q_0, a, b) - (q_0, \epsilon)$

no operation:- $\delta(q_0, a, b) - (q_0, b)$

$Z_0 \to$ stack start symbol

$M = (Q, \Sigma, \Gamma, Z_0, \delta, q_0, F)$

Q set of states, Σ input symbol, Γ stack symbol, Z_0 initial stack symbol, δ transition f^n, q_0 initial state, F final state

Γ — push, pop

$\delta: (Q \times \Sigma \cup \{\epsilon\} \times \gamma) \to (Q \times \gamma^*)$

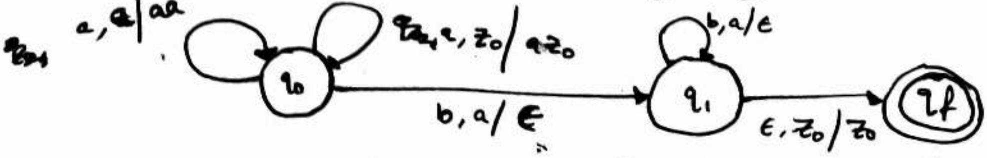
Q. Construct a PDA for the language $a^n b^n$ where $n \geq 1$

For every 'a', push 'a' in the stack

for every 'b', pop 'b' from the stack

Remember :- Every valid string should reach the final state
Every invalid string should not reach the final state interesting

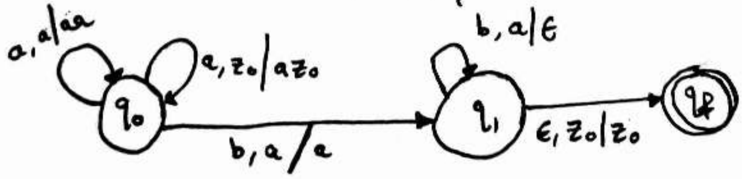
1. $\delta(q_0, a, z_0) - (q_0, az_0)$
2. $\delta(q_0, a, a) - (q_0, aa)$
3. $\delta(q_0, b, a) - (q_1, \epsilon)$
4. $\delta(q_1, b, a) - (q_1, \epsilon)$
5. $\delta(q_1, \epsilon, z_0) - (q_f, z_0)$



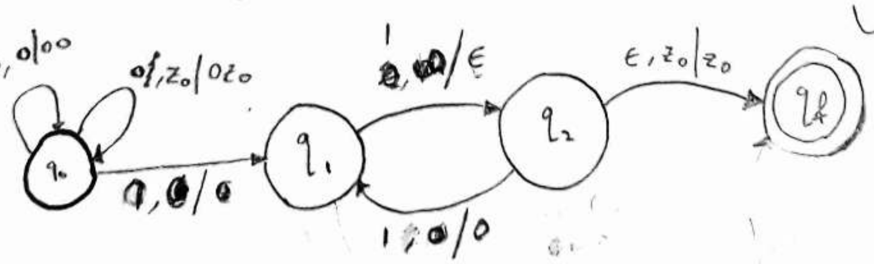
$Q = \{q_0, q_1, q_f\}$ $\Sigma = \{a, b\}$ $\Gamma = \{a, b, z_0\}$ $q_0 = \{q_0\}$
$F = \{q_f\}$ $Z_0 = \{z_0\}$

$L = \{a^m b^{m+n} c^n, n \geq 1\}$

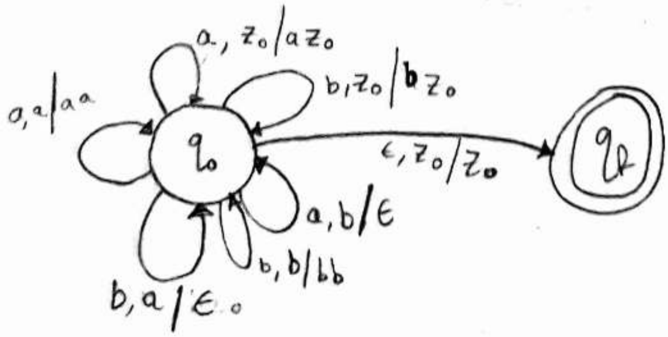
Q. Construct a PDA :- $L = \{a^n, b^{n+1}, n \geq 1\}$
$L = \{0^n 1^{2n}\}$
$L = \{$ equal no. of a's and equal no. of b's.



interesting
this is deterministic
00 1111



00 1111
000 111111



NP-complete → • they are in NP
• Any problem in NP can be reducible to this problem in P time complexity.

NP hard → (not point 1, only 2 is valid)

NP → • they can be solved in P time complexity using non-deterministic turing machine
• Problems that can be verified in P time complexity but not be solved in P time complexity using a deterministic turing machine.

PAGE NO.
DATE

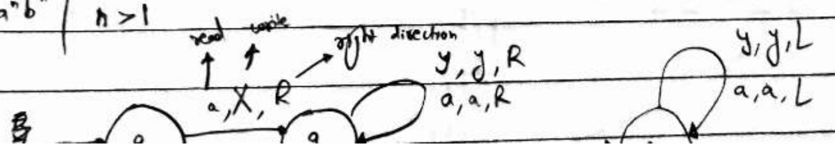
Turing m/c

for every successful string it will reach a final state qd
for every non successful string it won't reach the final state.

• was invented by 1936 by Alan Turing
• it is a device which accepts recursive enumerable language generated by type 0 grammar.
• there are various features of Turing m/c :-
  1. it has an external memory which remembers arbitary long sequence of input.
  2. it has unlimited memory capability.
  3. the model has a facility by which the input at left or Right on the tape can be read easily.
  4. the machine can produce certain output

Turing m/c does not have a 'ε' symbol

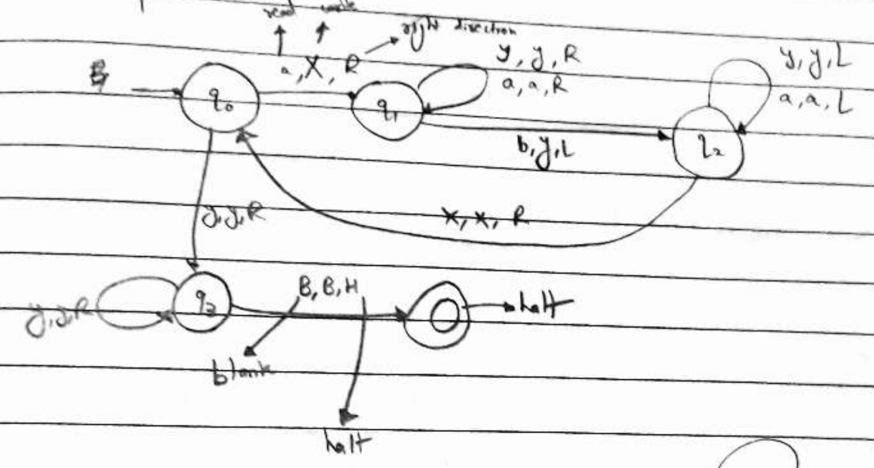
Q. $L = a^n b^n \mid n > 1$



produce certain output

Turing m/c does not have a 'ε' symbol

Q $L = a^n b^n \mid n \geq 1$



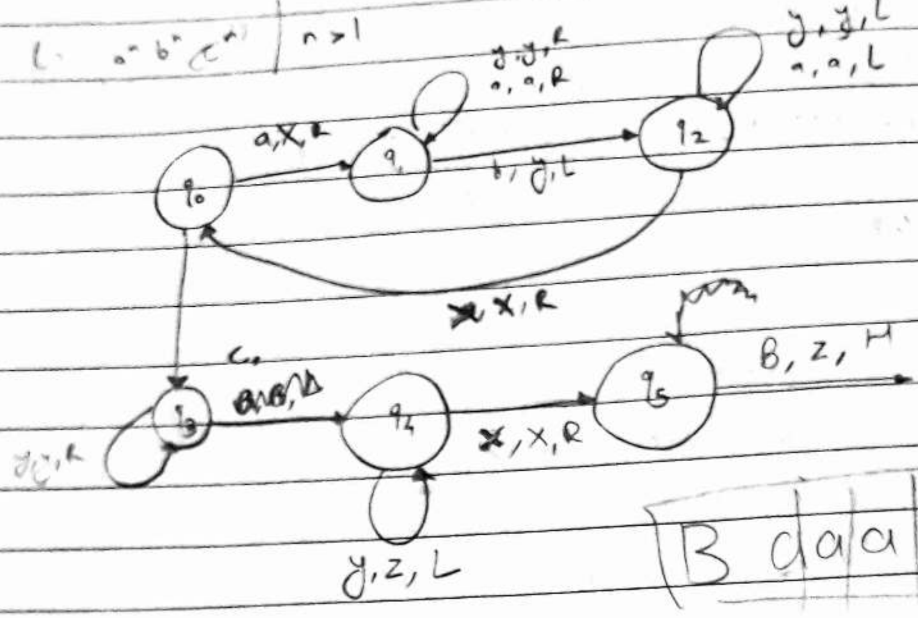
each cycle will consist of

1. leftmost 'a' is changed to 'X'
2. the first 'b' is changed to 'Y'
3. have come back to first 'a'

interesting gist

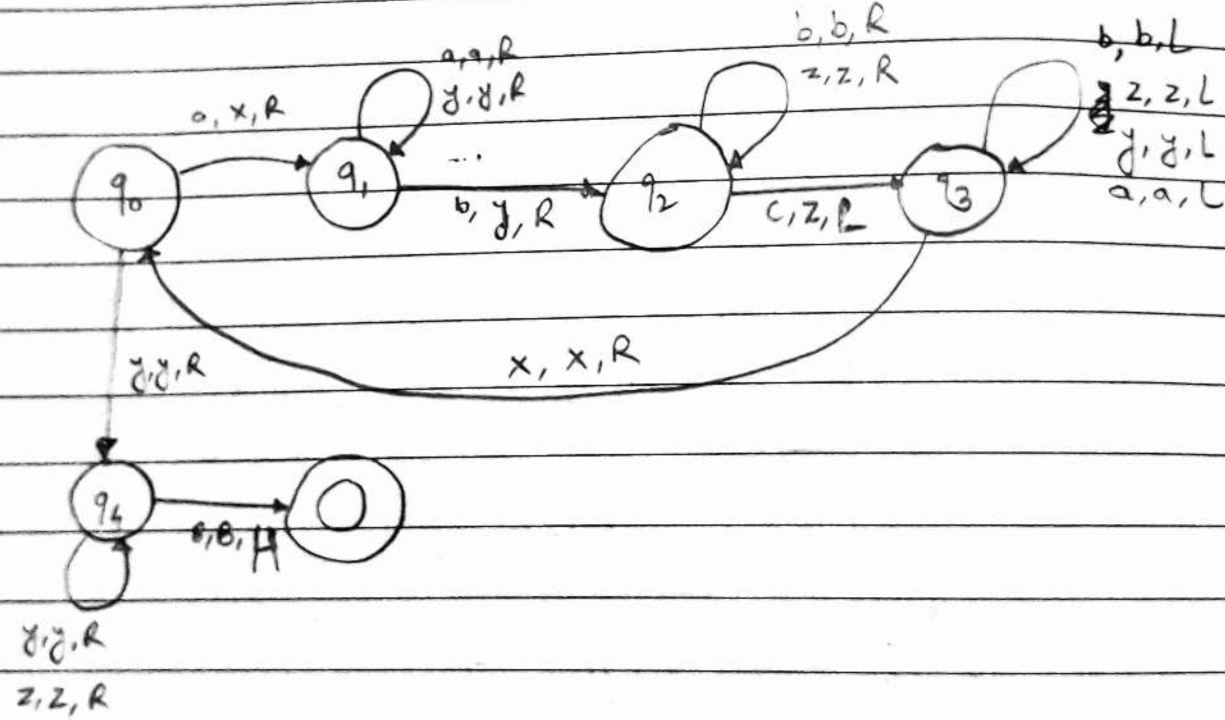
For every successful string, there is a state change but for unsuccessful string, there can be no movement from one state. It is called dead configuration. We will halt in that non final state and say that string is in non-final state.

Q. $L = a^n b^n c^n \mid n \geq 1$



wrong

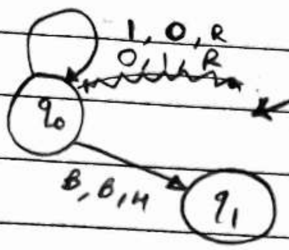
B	a	a	a	b b b c c c	B



[Input is not coming, it is already in the tape]

interesting — For transducers and generator, there is no final state. ($F = \phi$ (null) for such states.)

Q. Generator of 1's complement



No final state as a generator, so inputs will be coming

Q. Generator for 2's complement

Q. Adding two 1 unary no's

B	0	0	0	#	0	0	B	B

soln:— remove #, last 0 will be B

$$m = (\Sigma, Q, q_0, F, \Gamma, B, \delta)$$

$\Sigma \rightarrow$ input

$Q \rightarrow$ states

$q_0 \rightarrow$ initial state

$F \rightarrow$ final state

$\Gamma \rightarrow$ tape symbol

$B \rightarrow$ blank symbol

$\delta : Q \times \Gamma \rightarrow (Q, \Gamma, (L, R))$

$a^n b^n c^n \longrightarrow$ acceptor

generator $\longrightarrow$ 1's complement

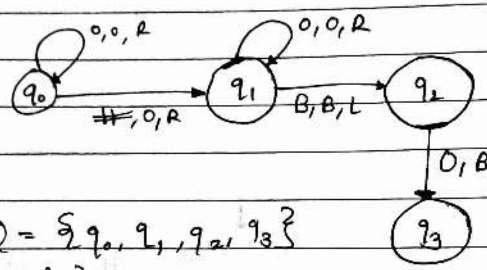
$\longrightarrow$ 2's complement

transducer $\longrightarrow$ mathematical symbol / operations

adding two unary no.s

$\Sigma = \{0\}$

B	0	0	0	#	0	0	B



7.22
8.39 } 1.17
9.15 } 0.76
9.56 } 0.41
9.56 } 0

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0\}$

$\Gamma = \{0, \#, B\}$

$\delta : Q \times \Gamma \rightarrow (Q, \Gamma, (L, R))$

$q_0 = \{q_0\}$

$F = \phi$ (as transducer)

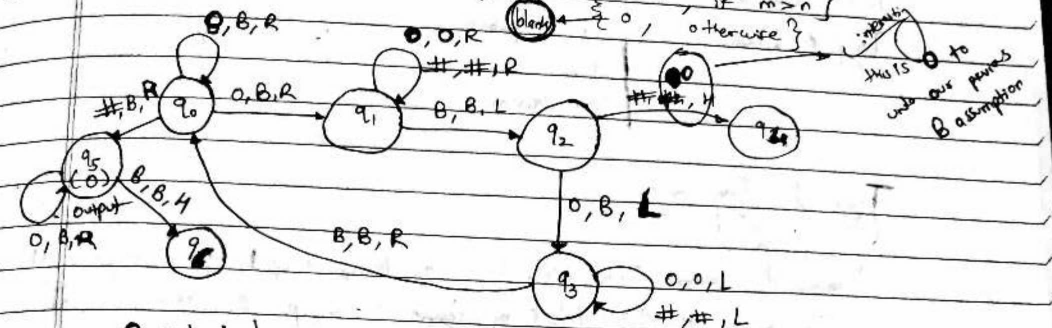
intensity

	0	#	B
q_0	$\{q_0, 0, R\}$	$\{q_1, 0, R\}$	—
q_1	$\{q_1, 0, R\}$	—	$\{q_2, B, L\}$
q_2	$\{q_2, B, H\}$	—	—
q_3	—	—	—

PAGE No.
DATE / /

Q. Design a turing machine to compute proper subtraction of 2 using no.s.
The proper sub. f^n is designed as follows:-

$$f(m,n) = \begin{cases} m-n, & \text{if } m>n \\ 0, & \text{otherwise} \end{cases}$$



B | B | B | B | 0 | B | B | B | B | B

B | B | B | B | # | 0 | B | B | B | B

B | B | B | 0 | 0 | B | B | B | B

B | B | B | # | 0 | 0 | B | B | B

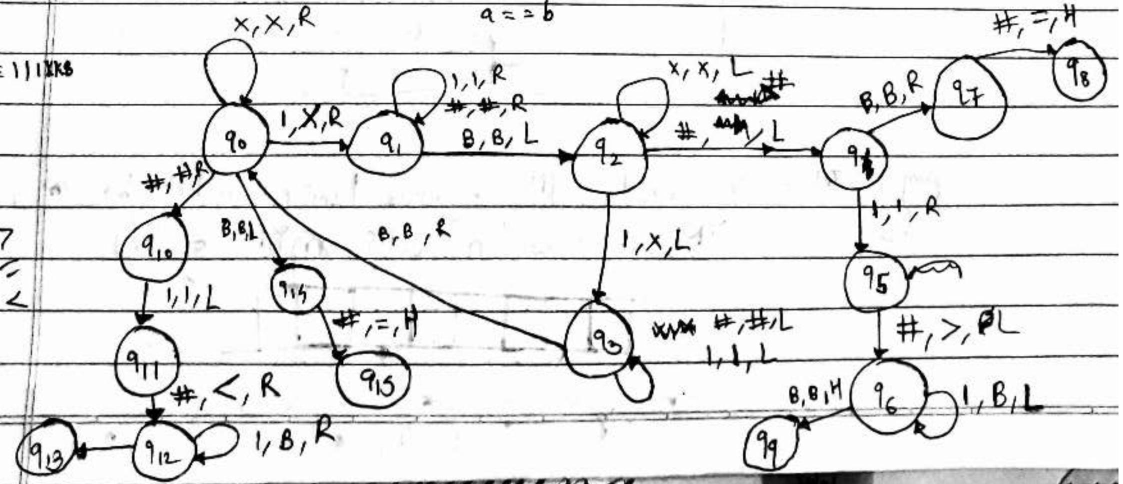
Q. Comparator a b

B 1111 # 111 B

a > b
a < b
a == b

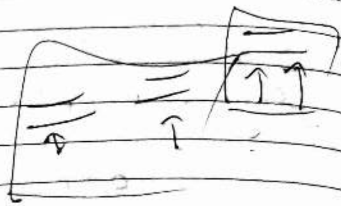
#111XXB

> = <



Context free language
Undecidability

Q. Multiplication using turing machine transducer



Types of Turing m/c	Undecidability
Power of Turing m/c	REL & RL
Turing ~~thesis~~ thesis	Halting problem
	PCP
	Rice Theorem

Types of Turing m/c

Variations of standard Turing m/c. No. of language accepted by a machine determines the power of that m/c & by changing some parameters (extra special characters). If we modify the TM, then also the power of TM remains same.

TM with stay option :— here there is extra stay movement apart from left or right movement.

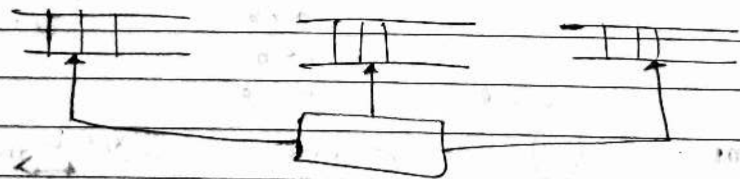
$$\delta : Q \times \gamma \rightarrow (Q, \gamma, (L, R, S))$$

TM with multitape movement :— it has more than 1 tape and more than one read, write head.

(can help provide parallelism)

$$\delta : Q \times \gamma^n \rightarrow (Q, \gamma^n, (L, R)^n)$$

n = no. of tape



jumping TM :— in standard TM, it moves 1 cell right or left but in jumping TM, it moves 'n' cells right or left.

$$\delta : Q \times \gamma \rightarrow (Q, \gamma^n, (L, R), n)$$

multi-dimensional → up, down, left, right
multi-head → one tape can have multiple heads

non erasing TM: here we cannot change any input to B (blank). So to change any input, don't write it as blank, but write any other symbol.

$$\delta : Q \times \gamma \rightarrow (Q, \gamma, (L,R))$$

Universal Turing machine →

Turing Thesis :
- Alan Turing in 1936 set a statement, till now no one has proved that it is wrong. No proof is there either to say it is right. General concensus is that he is right.

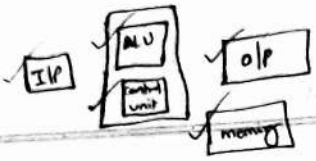
Points in Turing thesis :-
- Anything that can be done by existing computer, can also be done by TM.
- No one has yet been able to suggest a problem solvable by algorithm for which a TM program cannot be written.
- Alternative models have been proposed for mechanical computation but none of them are more powerful than the TM model.
  eg:- Lambda calculus

A computer can solve any problem but a TM can solve only a specific problem for which it is designed. In order to prove that Turing Thesis is correct, Alan Turing came up with the idea of Universal Turing m/c.

Von Neumann vs Harvard

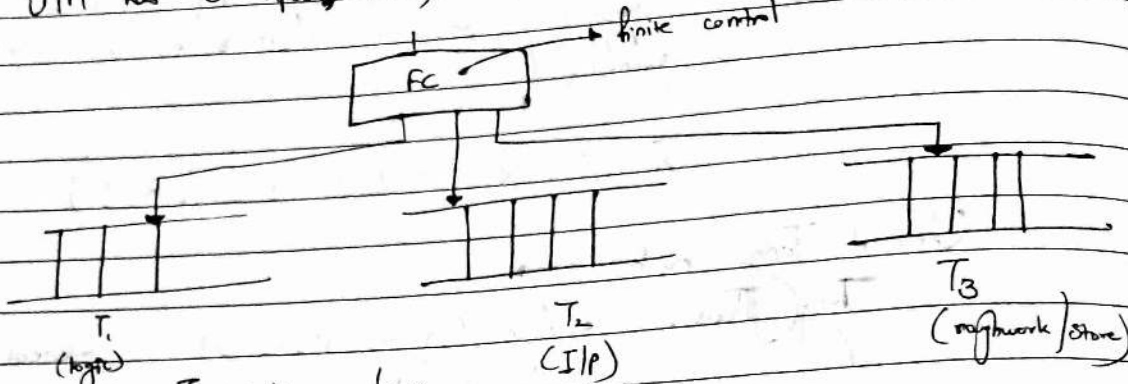
code and data stored in same way

separate code and bus pathways for code and data



Similarity with Von Neumann Architecture

UTM has 3 tapes :- T_1, T_2, T_3.



for example :- T_1 contains logic

T_2 contains the I/P

T_3 contains the roughwork or current internal states.

Let there are set of state $Q = q_0, q_1, q_2, q_3 \ldots$ and so on.

and $\gamma = \{a_1, a_2, a_3, \ldots\}$

now encode every symbol :- for eg:-

$q_0 = 1$

$q_1 = 11$

$q_2 = 111$

similarly eg:-

$a_1 = 1$

$a_2 = 11$

$a_3 = 111$

and so on

eg:- aabb → 1 | 1 | 0 | 1 | 1 | 1 | 1

given a = 1

b = 11

0 works as separator.

Now a UTM can represent a transition

eg:- $(q_1, a) \rightarrow (q_2, b, L)$

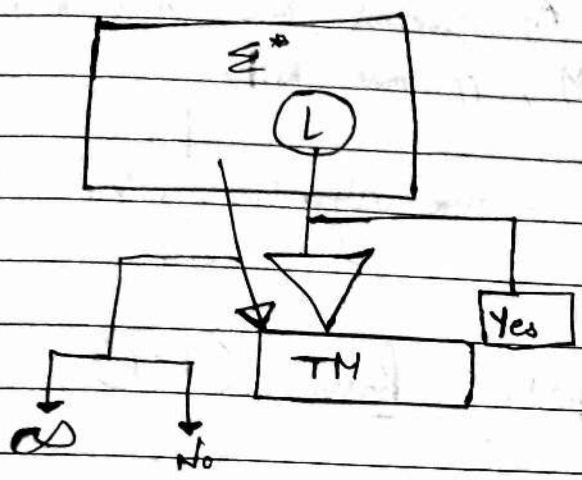
0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1

$\underbrace{11}_{q_1}$ $\underbrace{1}_{a}$ $\underbrace{111}_{q_2}$ $\underbrace{11}_{b}$ $\underbrace{1}_{L}$

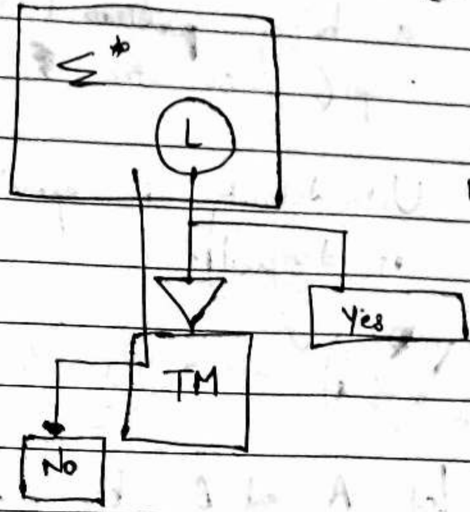
L = 1

UTM can be represented by a string of 0's and 1's. UTM act on T_1 tape by seeing input from Tape T2 and output can also return on T_2. So UTM act as computer.

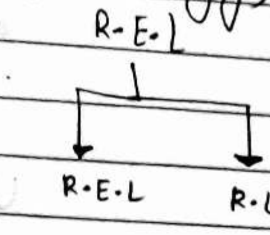
Undecidability



recursive enumerable language
(R.E.L)



recursive language
(R.L)



R.E.L (Turing recognizable language)

- A language accepted by T.M is called recursive enumerable language where L is accepted by TM. When L is given to TM, it says yes.
- If the language that is not present and is given to T.M it may say 'No' or enter in an infinite loop.
- So we will not be able to know whether TM will be able to say 'No' or enter an infinite loop.

R.L (Turing decidable language)

- A TM that says either 'yes' or 'no' without falling in infinite loop. Such time of TM is called halting TM.
- It says 'yes' if present in L
  Else says 'no' if not in L

Decidability and Undecibality

Both computability and decidability are same. That there exists an algorithm or not for a set of problem.

If for a problem, there exist an algorithm, then there exists a halting ~~problem~~ TM. As algorithm halts after a finite no. of steps, so for TM, it must halt.

Undecidability is equivalent to those algorithms for where normal TM is designed.

(check) • will PCP have a unique sol^n

Post Correspondence Problem (PCP)

Let A and B be two non empty set of string over Σ given as below:-

$$A = \{x_1, x_2, x_3, x_4 \dots x_k\}$$
$$B = \{y_1, y_2, y_3, y_4 \dots y_k\}$$

There is a post correspondence between A and B if there is a sequence $i, j, k, \dots m$ such that the string

$$x_i x_j x_k \dots x_m = y_i y_j y_k \dots y_m$$

Q. Does the ~~post~~ correspondence with 2 lists

$$A = \{a, aba^3, ab\}$$
$$B = \{a^3, ab, b\}$$

	1	2	3
A_i	a	aba^3	ab
B_i	a^3	ab	b

2, 1, 1, 3

	2	1	1	3
A_i	aba^3	a	a	ab
B_i	ab	a^3	a^3	b

PAGE No.
DATE / /

2 1 1 3

a b aaa aa ab
ab aaa aaab → 2, 1, 1, 3

→ usually a soln (for [illegible] → usually a soln)

Q. determine the soln for following instance of PCP

	list A	list B
	w_i	x_i
1	01	0
2	110010	0
3	1	1111
4	11	01

	1	2	3	4
w_i	01	110010	1	11
x_i	0	0	1111	01

1 2 ~~4~~ 1 1

0 11 ~~1~~ 1 01 0 10

0 1 1 11 01 0 0

$\{1, 3, 2, 4, 4, 3\}$

3 4 3 1 4 2

1 11 1 01 11 110010

1111 01 1111 0 01 0

Q. Does PCP with 2 list :-

A = { 10, 011, 101 }

B = { 101, 11, 011 }

1	2	3
10	011	101
101	11	011

1 1 3 3 3

~~1010 101~~

~~1010 1011~~

1, 3, 3, 3 . . . .

the PCP goes to infinite implementation of 3,

so it has no soln. But with any machine we cannot decide that it halts and say no.

Roughwork

~~10 101 101~~

~~101 011 011~~ . . .